▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación

12

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 12

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.

Diseño y maquetación:
BRAVO-LOFISH.

Fotografía:
EQUIPO GALATA.

Dibujos:
JOSE OCHOA

TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de AULA DE INFORMATICA APLICADA (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomuniación y José Luis Tojo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: (desde el tomo 5): Juan José Gómez, Licenciado en Química. LOGO: Cristina Manzanero, Licenciada en Informática. APLICACIONES: Soledad Tamariz, Diplomada en Telecomunicación. OTROS LENGUAJES (sistemas operativos): Domingo Villaseñor, Diplomado en Informática, y Lenguaje C: Enrique Serrano, Ingeniero en Telecomunicación.

Ediciones Siglo Cultural, S.A.

Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.

Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-104-5.
ISBN de la obra: 84-7688-068-7

Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.

Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M-5-677-1987
Printed in Spain - Impreso en España.

Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.

Mayo, 1987.
P.V.P. Canarias: 335,–.

INDICE

INFORMATICA BASICA





Almacenamiento interno. Registros y ficheros

TENIENDO en cuenta que la misión más importante de un ordenador es el tratamiento mecanizado de la información y que los computadores no trabajan sólo con datos individuales, sino con conjuntos de datos lógicamente relacionados, es necesario que dichos conjuntos de datos posean cierta estructura lógica y que se almacenen de acuerdo a otra estructura, la estructura física.

La organización de la información en el momento de su almacenamiento no sólo facilita la tarea del programador, sino que también aumenta la eficiencia en la gestión por parte del ordenador.

Los principales conceptos necesarios para la organización lógica de la información son: *el fichero* y *el registro.*



Registros y ficheros

Aquellos conjuntos a los que nos referíamos anteriormente es lo que se considera como *fichero.* Quizá el concepto de fichero provenga de los que se emplean en cualquier oficina para almacenar la información referente a todos los clientes.

Llamamos *registro* a un conjunto de informaciones referentes a un mismo tema y que, por tanto, constituyen una unidad de procesamiento. Dentro de cada registro se puede hacer una división en unidades de tratamiento como entidad propia dentro del registro. A estas unidades se las denomina *campos.*

Para identificar cada registro, existe en él un campo llamado *clave del registro,* que sirve de indicativo para distinguirlo de cualquier otro dentro del mismo fichero.

En cuanto a los ficheros, se distinguen





Jerarquía de almacenamiento interno de los datos.

entre sí por su nombre físico, aunque podrán tener asignados también un nombre lógico por el que serán accedidos desde cualquier programa.

Los ficheros pueden ser multivolumen cuando necesitan más de un soporte de información para su almacenamiento o, en caso contrario, los soportes se llaman multifichero cuando disponen de suficiente espacio para almacenar más de un fichero.

Refiriéndonos a registros, podemos distinguir los llamados *registros lógicos* de los *registros físicos*. Los primeros suelen conocerse simplemente como registros, y son conjuntos coherentes de informaciones agrupadas (el conjunto de informaciones de un cliente, de un artículo, etcétera), es decir, tienen por objeto agrupar un conjunto de campos que hacen referencia a un mismo concepto. En cambio, un registro físico es la unidad de información que un soporte lee o graba de una sola vez, y será un bloque de cinta, un sector de disco, etc. Un registro físico es una agrupación de registros lógicos, de esta forma se consigue disminuir el número de operaciones de entrada/salida, ya que en cada una de ellas se leerán o escribirán todos los registros lógicos contenidos en el mismo bloque (es muy frecuente llamar bloque al registro físico).

Registros de 200 bytes
Bloque de 1.024 bytes
200
1.024 VARIOS REGISTROS/BLOQUE

Registro de 1.000 bytes
Bloques de 1.024 bytes
1.000
1.024

Registro de 2.000 bytes
Bloque de 1.024 bytes
1.024
2.000

Normalmente los lenguajes de programación de alto nivel poseen instrucciones de lectura y grabación de registros, encargándose el computador de que el programa objeto genere las necesarias lecturas o grabaciones de bloques, liberando al programador de tener en cuenta la diferencia entre registro y bloque en cada dispositivo.



Formatos de registros y campos

No es necesario que todos los registros de un mismo fichero posean el mismo formato; existen cinco formas básicas de organizar los registros de un fichero.

RECFM	AREA DE DATOS	OTROS PARAMETROS
F	BLOQUE=REGISTRO	BLKSIZE
FB	REG-1 REG-2 REG-M BLOQUE	LRECL BLKSIZE=M. LRECL
V	LONG. DEL BLOQUE | BLOQUE=REGISTRO	BLKSIZE=TAMAÑO MAX. + 4
VB	REG. REG. REG. LONG. BLOQUE LONG. REG. DATOS	BLKSIZE Y LRECL BLKSIZE=4+M. LRECL LRECL=4+max. tamaño registro lógico
U	BLOQUE=REGISTRO	BLKSIZE=MAX. LONG. DE UN BLOQUE



Parámetros necesarios para definir el área de datos en un registro.

Como hemos dicho anteriormente, dentro de un mismo registro estarán contenidas ciertas informaciones, cada una de las cuales será un *campo* del registro. Cada uno estos campos puede estar dividido a su vez en otros subcampos o grupos de datos relativos a un concepto; por ejemplo, si tenemos un registro de un libro: título, autor y referencia serían campos del registro, mientras que la referencia podría contener dos subcampos: área a la que pertenece y localización dentro del área.





Estructura del registro LIBRO.

Los campos contenidos en un registro pueden ser de longitud variable o fija,

análogamente a los registros de un fichero. La gestión de los tamaños, tanto de los registros como de los campos, es responsabilidad del programa de usuario. Realmente es más fácil trabajar con ficheros cuyos registros sean todos de la misma longitud y que a su vez contengan campos del mismo tamaño. Al diseñar un registro se pueden plantear problemas para definir su longitud por varias causas:

— Campos de longitud variable: se puede tomar la máxima longitud posible como tamaño del campo, en cuyo caso se desaprovecha mucho espacio. En el caso de informaciones que lo permitan se pueden buscar abreviaturas y tomar un tamaño menor al máximo posible.

— Campos con un número variable de subcampos: como el caso anterior, si se tomase el mayor se desaprovecharía mucho espacio.

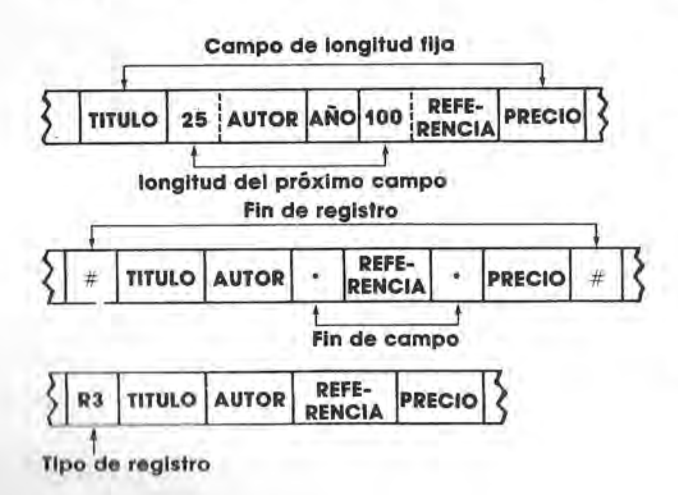
— Campos no existentes en todos los registros: en el caso en que sólo un número pequeño de registros tuviesen estos campos «extras», se desaprovecharía igualmente el soporte.

Cuando se utilizan registros de longitud variable, el manejo se hace muy difícil en programación, a no ser que el sistema operativo o el lenguaje de programación conozcan su longitud previamente. Existen diversos métodos para que el programa, al leer un registro sepa cuál es su longitud.

— Colocar delante de cada campo su longitud.

— Utilización de símbolos especiales de fin de campo y fin de registro.

— Incluir un campo en la primera posición que indique la configuración y/o tamaño del registro.





Tipos de ficheros

Las principales características de un fichero y por las cuales se podría hacer una clasificación son las siguientes:

1. **Tamaño**

Calculado en base al tamaño de sus registros y campos.

2. **Volatilidad**

Es decir, el tiempo que la información va a permanecer sin modificarse en el fichero.

3. **Actividad**

La frecuencia con que el fichero es referenciado indica la actividad que tiene. Así se pueden distinguir:

A. *Ficheros permanentes*

Son ficheros que no desaparecen por muchos accesos que se hagan a ellos. Pueden ser tratados y consultados cualquier número de veces.

Pueden ser:

— *Constantes*

Su información no se altera nunca o casi nunca por la naturaleza de los datos que contienen. Por ejemplo, un fichero que contenga la tabla de cosenos (invariable), o un fichero que contenga los libros que contiene una biblioteca (poco variable).

— *De situación*

También llamado FICHERO MAESTRO. Contienen en cada momento la información actualizada. Cualquier variación es registrada en ellos con la máxima urgencia.

— *Históricos*

Contienen información acerca de situaciones ya pasadas. Suelen utilizarse para establecer estadísticas.

B. *Ficheros de movimiento*

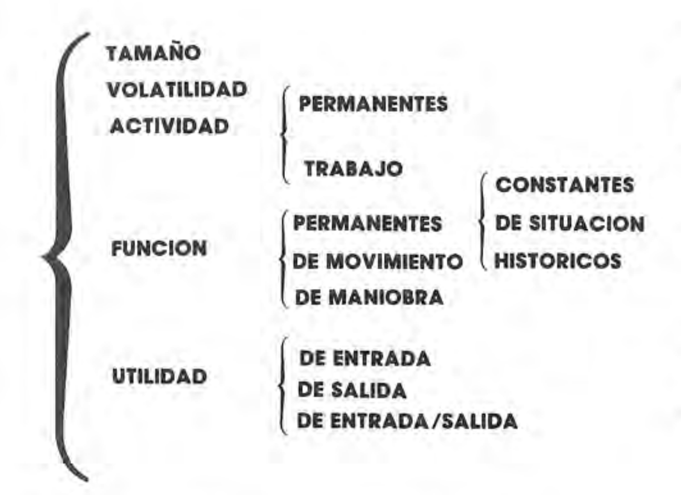
También se conocen como ficheros de transacciones y contienen los datos necesarios para actualizar o consultar un fichero permanente.

C. *De maniobra*

O ficheros de trabajo, son los que el ordenador crea con los resultados intermedios de un proceso, para reutilizarlo inmediatamente por el mismo proceso.

4. **Según su utilidad**

— Archivos de entrada. Son aquellos

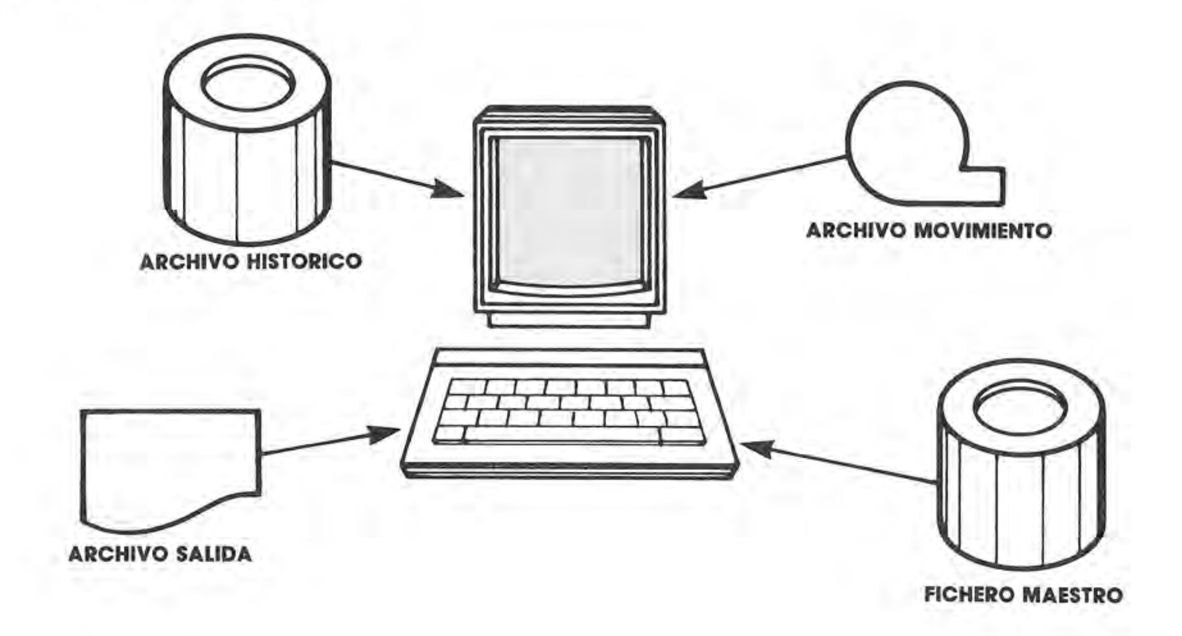



Cuadro clasificación de los ficheros.

que se utilizan para introducir en la memoria del ordenador la información que contienen. También se denominan *archivos de origen.*

— Archivos de salida. Se utilizan para almacenar la información extraída de la memoria interna del ordenador; también se llaman *ficheros de destino.*

— Archivos de entrada/salida. Se emplean tanto como ficheros de origen como en funciones de destino de la información procesada por el ordenador.





MAQUINA Z-80

SPECTRUM, AMSTRAD, MSX
OPERACIONES CON 16 BITS

COMO ya tratamos en el apartado anterior, el Z-80 puede realizar operaciones aritméticas con palabras de 16 bits de una forma simultánea. Esto supone un rango numérico de 0-65535 (2^{16}) que puede ser suficiente en muchos casos. De todas formas, la precisión así obtenida es deficiente en operaciones con coma flotante, por lo que será necesario la realización de operaciones con un ancho de palabra mayor (a veces llamado mantisa). Esto tendrá que hacerse con varias instrucciones utilizando el bit de carry (CY) o acarreo, para guardar lo que «llevamos» de la operación anterior. En la tabla adjunta podemos ver las operaciones disponibles de 16 bits.

Grupo aritmético de 16 bits

Código mnemotécnico	Operación simbólica	Indicadores S Z H P/V N C	Códigos 76 543 210 Hex	N.º de bytes	N.º de ciclos M	N.º de estados T	Comentarios
ADD HL,ss	HL←HL+ss	• • X X X • 0 ↕	00 ss1 001	1	3	11	ss Par
ADC HL,ss	HL←HL+ss+CY	↕ ↕ X X X V 0 ↕	11 101 101 ED 01 ss1 010	2	4	15	00 BC 01 DE 10 HL
SBC HL,ss	HL←HL−ss−CY	↕ ↕ X X X V 1 ↕	11 101 101 ED 01 ss0 010	2	4	15	11 SP
ADD IX,pp	IX←IX+pp	• • X X X • 0 ↕	11 011 101 DD 01 pp1 001	2	4	15	pp Par 00 BC 01 DE 10 IX 11 SP
ADD IY,rr	IY←IY+rr	• • X X X • 0 ↕	11 111 101 FD 00 rr1 001	2	4	15	rr Par 00 BC 01 DE 10 IY 11 SP
INC ss	ss←ss+1	• • X • X • • •	00 ss0 011	1	1	6	
INC IX	IX←IX+1	• • X • X • • •	11 011 101 DD 00 100 011 23	2	2	10	
INC IY	IY←IY+1	• • X • X • • •	11 111 101 FD 00 100 011 23	2	2	10	
DEC ss	ss←ss−1	• • X • X • • •	00 ss1 011	1	1	6	
DEC IX	IX←IX−1	• • X • X • • •	11 011 101 DD 00 101 011 2B	2	2	10	
DEC IY	IY←IY−1	• • X • X • • •	11 111 101 FD 00 101 011 2B	2	2	10	

NOTAS: ss es cualquiera de los pares de registros BC, DE, HL, SP.
pp es cualquiera de los pares de registros BC, DE, IX, SP.
rr es cualquiera de los pares de registros BC, DE, IY, SP.

Expliquemos brevemente cada una de ellas:

— La instrucción ADD HL,ss realiza la suma del contenido de los registros HL (16 bits), con los dos registros que indican ss, el resultado queda almacenado en los registros HL, afectando a los bits correspondientes del registro de flags, dependiendo del resultado.

— La instrucción ADC HL,ss realiza la

misma operación que la anterior, pero suma además el contenido del flag de acarreo (CY = 0,1) al bit menos significativo del registro L.

— La instrucción SBC HL,ss resta al contenido de HL el del registro indicado en ss y luego resta el contenido del bit de acarreo.

— Las instrucciones ADD IX,pp y ADD IY,rr suman a los registros IX e IY, los registros indicados por pp y rr, quedando los resultados en los registros IX e IY respectivamente.

— Las instrucciones INC HH suman 1 al registro de 16 bits indicado por hh, siendo éste cualquiera de esta longitud de los disponibles en el Z-80.

— Las instrucciones DEC hh realizan la operación contraria. Restan 1 a cualquier registro de 16 bits.

Como puede verse en la tabla, las operaciones con 16 bits son sólo posibles con registros, no con posiciones de memoria (direccionamiento inmediato, indirecto, etc.).



Operaciones de rotación y desplazamiento

Este tipo de instrucciones realizan una operación de gran utilidad, ya que no debemos olvidar que la multiplicación en binario equivale a desplazamientos y rotaciones reiteradas (multiplicar por 2 es lo mismo que desplazar un bit.)

En la tabla que sigue puede verse que el Z-80 permite casi todas las combinaciones en este tipo de instrucciones.

Grupo de rotación y desplazamiento

Código mnemotécnico	Operación simbólica	Indicadores S Z H P/V N C	Códigos 76 543 210 Hex	N.º de bytes	N.º de ciclos M	N.º de estados T	Comentarios
RLCA	CY ← 7←0 ← (A)	• • X 0 X • 0 ↕	00 000 111 07	1	1	4	Rotación circular a la izquierda del acumulador
RLA	CY ← 7←0 (A)	• • X 0 X • 0 ↕	00 010 111 17	1	1	4	Rotación a la izquierda del acumulador
RRCA	→ 7→0 → CY (A)	• • X 0 X • 0 ↕	00 001 111 0F	1	1	4	Rotación circular a la derecha del acumulador
RRA	→ 7→0 → CY (A)	• • X 0 X • 0 ↕	00 011 111 1F	1	1	4	Rotación a la derecha del acumulador
RLC r	CY ← 7←0 ← r,(HL),(IX+d),(IY+d)	↕ ↕ X 0 X P 0 ↕	11 001 011 CB 00 [000] r	2	2	8	Rotación circular a la izquierda del registro r
RLC (HL)		↕ ↕ X 0 X P 0 ↕	11 001 011 CB 00 [000] 110	2	4	15	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (IX+d)		↕ ↕ X 0 X P 0 ↕	11 011 101 DD 11 001 011 CB ←d→ 00 [000] 110	4	6	23	
RLC (IY+d)		↕ ↕ X 0 X P 0 ↕	11 111 101 FD 11 001 011 CB ←d→ 00 [000] 110	4	6	23	Todas con el mismo formato y estados que las RLC; el código se forma como en las RLC, pero reemplazando el [000] de RLC por el propio código de la operación.
RL m	CY ← 7←0 ← m≡r,(HL),(IX+d),(IY+d)	↕ ↕ X 0 X P 0 ↕	[010]				
RRC m	→ 7→0 → CY m≡r,(HL),(IX+d),(IY+d)	↕ ↕ X 0 X P 0 ↕	[001]				
RR m	→ 7→0 → CY m≡r,(HL),(IX+d),(IY+d)	↕ ↕ X 0 X P 0 ↕	[011]				
SLA m	CY ← 7←0 ← 0 m≡r,(HL),(IX+d),(IY+d)	↕ ↕ X 0 X P 0 ↕	[100]				
SRA m	7→0 → CY m≡r,(HL),(IX+d),(IY+d)	↕ ↕ X 0 X P 0 ↕	[101]				
SRL m	0 → 7→0 → CY m≡r,(HL),(IX+d),(IY+d)	↕ ↕ X 0 X P 0 ↕	[111]				
RLD	7-4 | 3-0 7-4 | 3-0 A (HL)	↕ ↕ X 0 X P 0 •	11 101 101 ED 01 101 111 6F	2	5	18	Rotación de las cifras BCD, a la derecha y a izquierda, entre la mitad baja del acumulador y la posición (HL); no afectan a la mitad alta del acumulador.
RRD	7-4 | 3-0 → 7-4 | 3-0 A (HL)	↕ ↕ X 0 X P 0 •	11 101 101 ED 01 100 111 67	2	5	18	

Expliquemos brevemente cada una de las instrucciones.:

— RLCA rota a la izquierda cíclicamente los bits del acumulador A. Independientemente de esta rotación cíclica (en la que *no* toma parte el bit de acarreo), el bit de acarreo toma el valor que tenía el bit 7 (más significativo) del acumulador, *antes* de ejecutarse la instrucción.

— RLA rota cíclicamente el acumulador a la izquierda, con el bit de acarreo.

— RRCA y RRA realizan las mismas operaciones que las instrucciones RLCA y RLA, respectivamente, pero la rotación es a la derecha.

— Como puede verse en la tabla, la instrucción RLC puede realizarse sobre cualquier registro de 8 bits (no sólo sobre A→RLCA, RLCB, etc.). Además, puede utilizarse el direccionamiento con el vector contenido en HL (RCL, (HL)), o también realizar un direccionamiento indirecto preindexado con IX o IY. (RLC (IX+d) o RLC (IY+d), como, por ejemplo, RLC (IX+$300F), en este caso se rotaría hacia la izquierda el contenido de la posición de memoria cuya dirección se encuentra en las posiciones IX+$300F y IX+$300F+1.

— Como puede verse en la tabla, la rotación de la izquierda permite también todas las posibilidades anteriores: RLm, donde *m* es un registro de 8 bits, (HL), (IX+d) y (IY+d).

— RRCm o RRm realizan las mismas operaciones que RLCm y RLm, pero la rotación es a la derecha.

Todas las instrucciones anteriores se referían a rotaciones cíclicas, es decir, los bits se desbordan por un lado y se inyectan por el opuesto.

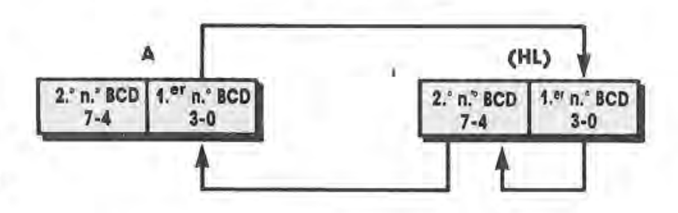
— Las operaciones SLAm y SRLm desplazan a la izquierda y derecha, respectivamente, los bits de *m*. (Los bits que desbordan van al bit de acarreo y el bit que falta se pone a cero.) *m* puede ser cualquier posibilidad de registro o direccionamiento de los explicados anteriormente.

— La instrucción SRAm funciona de una forma similar a SRLm, pero el bit 7, que falta por ser un desplazamiento a la derecha, no se pone a cero, sino que toma el mismo valor que tenía antes del desplazamiento, por lo que los bits 7 y 6 tomarán el mismo valor.

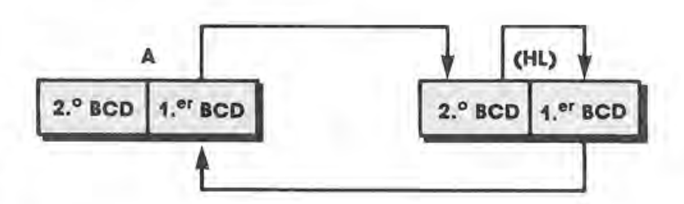
— La función RLD está indicada para la operación con números en BCD. En esta operación, los *n* bits menos significativos del byte cuya dirección se encuentra en HL (primer dígito BC de (HL)) son trasladados a la segunda posición de este mismo byte (bits del 4-7) y el contenido previo de estas cuatro posiciones se pasa a los *n* bits menos significativos el acumulador. El contenido de este primer dígito BCD del acumulador se pasa al primero del byte de la posición memoria (HL).

El segundo número BDC del acumulador no se ve afectado.

Ver el esquema adjunto.



— La instrucción RRD realiza la rotación contraria a la instrucción RLD, como se indica en la figura.



PROGRAMAS

EDUCATIVOS • DE UTILIDAD • DE GESTION • DE JUEGOS



Programa: La guarida del dragón

ESTE programa podemos englobarlo dentro de la categoría de programas conversacionales. El jugador tiene que superar una serie de dificultades para llevar a buen término la aventura.

Una aventura conversacional es aquella en la que le damos mensajes escritos al ordenador y él los entiende y ejecuta. Por ejemplo, le podemos decir al ordenador que coja una cosa del suelo, que la suelte, que se mueva hacia determinada dirección, que mire a su alrededor. Todo esto se lo decimos mediante palabras en español y de la forma que nos plazca.

Se ha intentado que la forma de introducir los comandos sea lo más sencilla y rápida posible. Por ello, todas las órdenes que se le den al ordenador han de tener la forma VERBO-OBJETO con todas las preposiciones que se quieran en medio. Por ejemplo, el programa nos admitiría mensajes como:

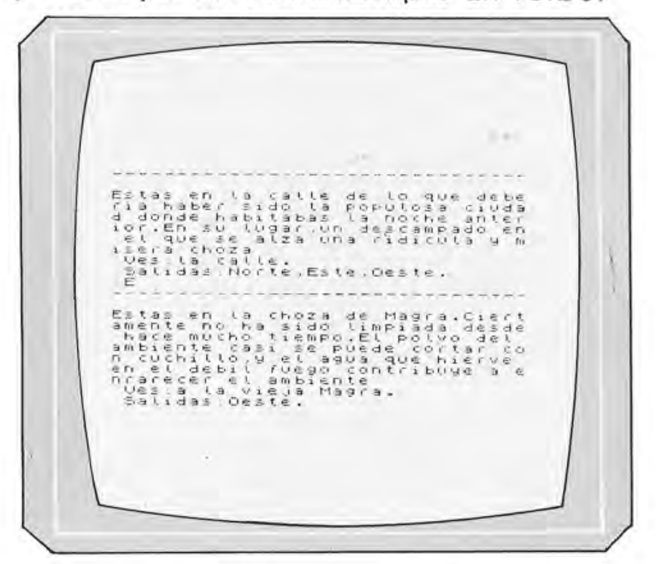
COGER EL CRISTAL
COGER CRISTAL
COGER UN CRISTAL
COGER JHJF EL UNO VARIOS ADIOS CRISTAL

Como puede apreciarse, lo importante es el verbo (que siempre ha de ir en infinitivo) y el objeto sobre el cual va a recaer la acción. Es imprescindible que la primera palabra sea siempre un verbo.

```
--------------------------------
Estas en tu casa.Sobre la cama h
ay una estatua de piedra caliza
y blanca.El pozo del agua tiene
la tapdera quitada.Todo parece a
bandonado,como si hubieras estad
o dormido siglos....
 Ves una estatua,el baul,tu casa
.
 Salidas Este.
>ABRIR BAUL
Que sorpresa,dentro hay Oro en
monedas y una linterna.

 AHORA QUE _
```

Primera pantalla del programa «La guarida del dragón» y la introducción de la primera orden «ABRIR BAUL».





Estás fuera de tu casa, ¿qué vas a hacer?

El programa, antes de procesar la entrada del usuario, comprueba que el objeto al que nos estamos refiriendo en la frase se encuentra en la habitación donde nos encontramos. Esto no pasa con los verbos IR y DECIR. Con estos verbos la acción se procesa directamente.

Otra particularidad es que el verbo IR no admite frases del tipo IR HASTA LA CASA. Con este verbo siempre habrá que utilizar estructuras como IR CASA. En el caso de que la dirección a la que nos queremos mover es uno de los cuatro puntos cardinales, tendremos que poner IR NORTE o IR SUR. En este caso específico podemos quitar la frase entera sustituirla por la primera letra de la dirección hacia la que nos queremos mover. Según esto, si queremos ir al Oeste sólo tendremos que teclear la letra O.

Los verbos que puede entender el programa son los siguientes:

- ENCENDER - Una linterna, una cerilla, la luz...
- IR
- ATACAR - A una persona o animal. Nunca objetos.
- MATAR - Igual que atacar.
- PONER - Ponerse ropa, poner alguna cosa en algún sitio.
- COGER - Coger objetos inanimados o plantas, nunca personas.
- DEJAR - Soltar algunas de nuestras pertenencias.
- EXAMINAR - Mirar algún objeto que esté en la habitación o que nosotros tengamos.
- ABRIR - Una puerta, un armario, una carta...
- LEER - Un papel, la pared...
- DECIR - Para hablar con alguien. El mensaje ha de ir entre comillas.
- DAR - Algo a alguien.

Otras palabras que también podemos utilizar son:

- DESCRIPCION - Para saber dónde estamos. Para echar una mirada a nuestro alrededor.
- INVENTARIO - Para saber qué objetos estamos llevando con nosotros.




Vas camino de un castillo, quién sabe lo que te espera.

El argumento del juego no lo decimos para que el jugador lo adivine mientras juega. Lo que sí podemos decir es que éste es uno de los mejores programas conversacionales que se han escrito en español y que su solución puede ser trabajo de varias semanas o meses.

```
 10>GO SUB 2600:GO SUB 3010
 20 DIM I(5): REM OBJETOS QUE LLEVA EL JUGADOR
 30 GO SUB 3150
 40 GO SUB 1020
 50 POKE 23658,8
 60 INPUT "Y AHORA QUE ? >"; LINE P$
 70 IF P$="" THEN  GO TO 60
 80 PRINT ">";P$
 90 LET P$=","+P$+","
100 LET FC=1
110 FOR I=FC+1 TO LEN P$
120 IF P$(I)="," THEN  LET LC=I: GO TO 140
130 NEXT I
```

```
 140 LET F$=P$(FC+1 TO LC-1)
 150 LET ERROR=0
 160 GO SUB 280
 170 IF ERROR THEN  GO TO 250
 180 IF MAPA=23 AND E(4,1)<>0 THEN  PRINT "Sin el flotador,te hundes en el agua
del foso.": GO TO 890
 190 IF MAPA=30 THEN  PRINT "Aaaarghh!!.Una trampilla se abrea tus pies precipit
andote por unprecipicio.": GO TO 890
 200 IF MAPA=6 AND E(15,3)=0 THEN  LET E(15,3)=1: PRINT "Al Emperador empteza a
no gus-  tarle tu presencia."
 210 IF mapa=6 AND e(15,3)>0 THEN  LET e(15,3)=e(15,3)+1: IF e(15,3)=5 THEN  PRI
NT "Los esbirros del Emperador te   han capturado.Ahora viviras en  la carcel el
 resto de tus dias.": GO TO 890
 220 IF mapa=6 AND e(15,3)<0 THEN  LET e(15,3)=e(15,3)-1: PRINT "El castillo se
derrumba!!!": IF e(15,3)=-4 THEN  PRINT "Has quedado sepultado en las    ruinas
del castillo.": GO TO 890
 230 IF e(12,3)=1 THEN  PRINT "Felicidades:conseguiste vencer  al emperador y de
volver la vida a la princesa.Mereces casarte   con ella.Que seais felices!!!.":
PAUSE 0: RUN
 240 IF MAPA=23 THEN  LET E(4,3)=E(4,3)+1: IF E(4,3)=3 THEN  PRINT "Los cocodril
os te estaban espe- rando para devorarte.Lo siento.": GO TO 890
 250 IF lc=LEN p$ THEN  GO TO 60
 260 LET fc=lc
 270 GO TO 110
 280 IF LEN f$=1 THEN  LET o$=f$: GO SUB 930: GO TO 720
 290 REM DETECCION DEL VERBO
 300 FOR I=1 TO LEN F$
 310 IF F$(I)=" " THEN  GO TO 340
 320 NEXT I
 330 LET V$=f$: GO TO 730
 340 LET V$=F$( TO I-1)
 350 LET O$=F$(I+1 TO )
 360 REM TRATAMIENTO VERBO
 370 RESTORE 2420
 380 LET VERBO=0
 390 READ A$
 400 IF A$="*" THEN  GO TO 770
 410 LET VERBO=VERBO+1
 420 IF LEN A$>LEN F$ THEN  GO TO 390
 430 IF F$( TO LEN A$)=A$ THEN  GO TO 450
 440 GO TO 390
 450 REM TRATAMIENTO OBJETO
 460 IF VERBO=13 THEN  GO SUB 2250: RETURN
 470 IF VERBO=3 THEN  GO SUB 1210: RETURN
 480 LET OBJETO=0
 490 IF LEN O$=1 THEN  GO SUB 930: GO TO 720
 500 RESTORE 2570
 510 READ A$
 520 IF A$="*" THEN  GO TO 770
 530 LET OBJETO=OBJETO+1
 540 IF LEN A$>LEN O$ THEN  GO TO 510
 550 FOR I=1 TO LEN O$-LEN A$+1
 560 IF O$(I TO I+LEN A$-1)=A$ THEN  GO TO 590
 570 NEXT I
 580 GO TO 510
 590 REM COMPRUEBA SI EL OBJETO ESTA O NO ESTA
 600 FOR I=1 TO 5
 610 IF I(I)=OBJETO THEN  GO TO 640
 620 NEXT I
 630 IF E(OBJETO,1)<>MAPA OR E(OBJETO,2)=0 THEN  PRINT "No lo veo por ninguna pa
rte.": RETURN
 640 IF VERBO=1 OR VERBO=2 THEN  GO SUB 2370
 650 IF VERBO=12 OR VERBO=4 THEN  GO SUB 1330
 660 IF VERBO=5 THEN  GO SUB 1640
 670 IF VERBO=6 THEN  GO SUB 1750
 680 IF VERBO=7 OR VERBO=8 THEN  GO SUB 1880
 690 IF VERBO=9 THEN  GO SUB 1950
 700 IF VERBO=10 THEN  GO SUB 2020
 710 IF VERBO=11 THEN  GO SUB 2120
```

```
 720 RETURN
 730 REM VERBOS SIN OBJETO
 740 IF LEN V$<3 THEN  GO TO 770
 750 IF V$( TO 3)="INV" THEN  GO SUB 810: RETURN
 760 IF V$( TO 3)="DES" THEN  GO SUB 1020: RETURN
 770 REM PALABRA NO RECONOCIDA
 780 PRINT "No entiendo."
 790 LET ERROR=1
 800 RETURN
 810 REM INVENTARIO
 820 LET t=0: PRINT "Llevas:";
 830 FOR I=1 TO 5
 840 IF I(I) THEN  LET T=1: LET OBJETO=I(I): GO SUB 1370: PRINT A$;",";
 850 NEXT I
 860 IF NOT T THEN  PRINT "nada ";
 870 PRINT CHR$ 8+"."
 880 RETURN
 890 REM MUERTE DEL PROTAGONISTA
 900 PRINT "Estas muerto,pulsa una tecla    para volver a jugar."
 910 PAUSE 0
 920 RUN
 930 REM IR EN UNA DIRECCION N,S,E,O
 940 IF O$<>"N" AND O$<>"S" AND O$<>"E" AND O$<>"O" THEN  PRINT "Ir donde ?.": R
ETURN
 950 LET D=0
 960 IF O$(1)="N" THEN  LET D=1
 970 IF O$(1)="S" THEN  LET D=3
 980 IF O$(1)="E" THEN  LET D=5
 990 IF O$(1)="O" THEN  LET D=7
1000 IF L$(MAPA,D TO D+1)="00" THEN  PRINT "No hay salida en esa direccion.": RE
TURN
1010 LET mapa=VAL (l$(mapa,d TO d+1))
1020 PRINT "--------------------------------"
1030 IF (INT (mapa/5)<4 OR mapa=19) AND (e(2,3)=0 OR E(2,1)<>0) THEN  PRINT "Est
a demasiado oscuro,no veo!!!.": RETURN
1040 LET desc=VAL l$(mapa,9 TO )
1050 PRINT : PRINT "Estas en ";: RESTORE 3020: FOR I=1 TO DESC: READ A$: NEXT I:
 PRINT A$
1060 PRINT " Ves:";
1070 LET t=0
1080 FOR i=1 TO 20
1090 IF e(i,2)=1 AND e(i,1)=mapa THEN  LET t=1: LET objeto=i: GO SUB 1370: PRINT
 a$;",";
1100 NEXT i
1110 IF NOT t THEN  PRINT "nada ";
1120 PRINT CHR$ 8+"."
1130 IF L$(MAPA, TO 8)="00000000" THEN  RETURN
1140 PRINT " Salidas:";
1150 IF VAL l$(mapa, TO 2) THEN  PRINT "Norte,";
1160 IF VAL l$(mapa,3 TO 4) THEN  PRINT "Sur,";
1170 IF VAL l$(mapa,5 TO 6) THEN  PRINT "Este,";
1180 IF VAL l$(mapa,7 TO 8) THEN  PRINT "Oeste,";
1190 PRINT CHR$ 8+"."
1200 RETURN
1210 REM IR
1220 IF LEN o$<3 THEN  PRINT "Por favor,vuelve a escribir eso.": RETURN
1230 LET O$=O$( TO 3)
1240 IF MAPA=28 AND o$="ARB" THEN  LET MAPA=29: GO TO 1020
1250 IF MAPA=25 AND O$="POZ" THEN  LET MAPA=19: GO TO 1020
1260 IF MAPA=6 AND o$="VEN" THEN  LET MAPA=23: GO TO 1020
1270 IF MAPA=19 AND o$="PUE" AND E(9,3)=1 THEN  LET MAPA=1: LET e(9,1)=mapa: GO
TO 1020
1280 IF MAPA=19 AND o$="CAS" THEN  LET MAPA=25: GO TO 1020
1290 IF MAPA=1 AND o$="PUE" THEN  LET MAPA=19: LET e(9,1)=mapa: GO TO 1020
1300 IF MAPA=26 AND O$="CHO" THEN  LET MAPA=27: GO TO 1020
1310 LET O$=O$(1)
1320 GO TO 930
1330 REM EXAMINAR UN OBJETO
1340 GO SUB 1370
```

```
1350 PRINT b$;"."
1360 RETURN
1370 REM BUSCA EL NOMBRE DEL OBJETO EN LA TABLA
1380 RESTORE 1440
1390 FOR n=1 TO objeto
1400 READ a$,b$
1410 NEXT n
1420 RETURN
1430 REM DESCRIPCIONES DE LOS OBJETOS
1440 DATA "la daga","Es una espada corta y muy afilada"
1450 DATA "la linterna","es una lampara de aceite"
1460 DATA "el diamante","La luz que se refleja en sus facetas le da un brillo ma
gico"
1470 DATA "el flotador","Es un flotador de corcho"
1480 DATA "la llave","Es una llave dorada con aspecto de vieja"
1490 DATA "el cofre del hechizo","Este hechizo destruira al Empe- rador,pero sol
o si lo usas de  acuerdo con las instrucciones"
1500 DATA "el pergamino","En el dice:Instrucciones para el manejo del hechizo:ab
rir la tapa y pronunciar las palabras ORO ORO CHOCOLATE DEL LORO"
1510 DATA "el oro","Es solo oro"
1520 DATA "una puerta","Una puerta de madera de roble,  vieja pero solida,y tien
e una  buena cerradura"
1530 DATA "una ventana","Esta abierta!!!"
1540 DATA "el viejo arbol","En su interior vivio alguien"
1550 DATA "una estatua","Es curioso,esa estatua es muy  parecida a la princesa
Aifos,es mas yo diria que es exactamente igual a ella.En una de las cuen-cas de
sus ojos hay un diamante,la otra esta vacia"
1560 DATA "el esqueleto","el esqueleto del emperador"
1570 DATA "a la vieja Magra","Parece muy vieja,debe tener al  menos un siglo"
1580 DATA "al emperador","Sus ojos se fijan en ti y te   dice:preparate a morir
,entrome- tido"
1590 DATA "el baul","Es un viejo baul que has usado  en muchos viajes"
1600 DATA "una cerradura","Aunque esta muy oxidada,todavia puede funcionar"
1610 DATA "un dragon","Es uno de los dragones escupe-  fuego escapados del Zoo"
1620 DATA "tu casa","Todo parece como si llevara mu- cho tiempo sin limpiar,suci
o y  abandonado"
1630 DATA "la calle","Esta desierta"
1640 REM COGER UN OBJETO
1650 IF OBJETO>8 THEN  PRINT "Es demasiado pesado.": RETURN
1660 IF E(OBJETO,1)=0 THEN  PRINT "Ya lo llevas encima.": RETURN
1670 FOR i=1 TO 5
1680 IF i(i)=0 THEN  GO TO 1720
1690 NEXT i
1700 PRINT "LLevas demasiado."
1710 RETURN
1720 PRINT "De acuerdo,coges ";: GO SUB 1370: PRINT a$;"."
1730 LET i(i)=objeto: LET E(OBJETO,1)=0
1740 RETURN
1750 REM DEJAR UN OBJETO
1760 IF MAPA=19 THEN  PRINT "No lo puedes dejar en ningun si-tio,se te caeria al
 fondo del  pozo.": RETURN
1770 IF mapa=23 THEN  PRINT "No lo puedes dejar,se hundiria."
1780 FOR i=1 TO 5
1790 IF i(i)=objeto THEN  GO TO 1830
1800 NEXT i
1810 PRINT "No lo llevas encima."
1820 RETURN
1830 LET i(i)=0: LET e(objeto,1)=mapa
1840 PRINT "De acuerdo,dejas ";
1850 GO SUB 1370
1860 PRINT a$;"."
1870 RETURN
1880 REM MATAR A ALGUIEN
1890 IF MAPA=6 AND OBJETO=15 THEN  PRINT "No recuerdas lo que tenias que  hacer
para matar al Emperador?.": RETURN
1900 IF mapa=27 THEN  PRINT "No seas asesino.": RETURN
1910 IF mapa=17 AND objeto=18 AND e(1,1)=0 AND E(18,2)=1 THEN  PRINT "Muy bien.M
atas al dragon con tu daga.El pobre animalito no tienetiempo ni de enterarse y m
uere  con un resoplido que hace tem-  blar ls paredes de la caverna.Ensus fauces
```

```
abiertas aparece un  cofre.": LET e(6,2)=1: LET E(18,2)=255: RETURN
1920 IF mapa=23 THEN  PRINT "Es inutil,son mas fuertes que tu.": RETURN
1930 IF MAPA=17 AND OBJETO=18 AND E(18,2)=255 THEN  PRINT "No puedes matarlo otr
a vez.": RETURN
1940 PRINT "No puedes hacer eso.": RETURN
1950 REM ENCENDER ALGO
1960 IF OBJETO<>2 THEN  PRINT "Eso no puedo encenderlo.": RETURN
1970 IF objeto=2 AND e(2,3)=1 THEN  PRINT "Ya esta encendida.": RETURN
1980 PRINT "De acuerdo,enciendes ";
1990 GO SUB 1370: PRINT a$
2000 LET e(2,3)=1
2010 RETURN
2020 REM ABRIR UN OBJETO
2030 IF e(8,2)=0 AND objeto=16 THEN  PRINT "Que sorpresa,dentro hay Oro en  mone
das y una linterna.": LET e(2,2)=1: LET e(8,2)=1: RETURN
2040 IF OBJETO=9 AND MAPA=19 AND E(9,3)=0 AND E(5,1)=0 THEN  PRINT "Abres la pue
rta del pozo.Al mo- verse sobre sus bisagras oxida- das produce un ruido siniest
ro.": LET e(9,3)=1: RETURN
2050 IF objeto=9 AND mapa=19 OR mapa=1 AND e(9,3)=1 THEN  PRINT "Ya esta abierta
.": RETURN
2060 IF OBJETO<>6 THEN  PRINT "No puedes.": RETURN
2070 IF mapa<>6 THEN  PRINT "Lo abres,pero se vuelve a cerrar": RETURN
2080 IF e(6,3)=1 THEN  PRINT "Ya esta abierto.": RETURN
2090 PRINT "El Emperador palidece cuando delcofre empieza a salir humo      blan
co."
2100 LET e(6,3)=1
2110 RETURN
2120 REM DAR
2130 IF MAPA<>27 OR OBJETO<>8 THEN  PRINT "No puedes hacer eso.": RETURN
2140 IF MAPA=27 AND OBJETO=8 AND E(5,2)=1 THEN  PRINT "La vieja te mira a los oj
os y secalla.": RETURN
2150 PRINT "La pobre vieja acepta encantada el oro que le ofreces y te dice:"
2160 PRINT " Hijo mio,hace siglos que estas dormido,desde que el malvado    Empe
rador de las Sombras lanzo  sobre el reino la maldicion que durmio a Aifos,la pr
incesa y"
2170 PRINT "trajo el hambre y la desolacion al reino.Debes acabar con el conel h
echizo que se encuentra en  la caverna del dragon."
2180 PRINT " La vieja Magra deja caer una   llave y dice:espero que la sepasusar
 "
2190 LET e(5,2)=1
2200 LET E(8,1)=255: LET E(8,2)=255
2210 FOR I=1 TO 5
2220 IF I(I)=8 THEN  LET I(I)=0
2230 NEXT I
2240 RETURN
2250 REM DECIR ALGO
2260 IF mapa=27 THEN  PRINT "La vieja esta sorda,no te puede oir.": RETURN
2270 IF mapa<>6 THEN  PRINT "Nadie oye lo que dices.": RETURN
2280 IF e(6,3)=0 THEN  PRINT "Lo dices,pero no pasa nada.": RETURN
2290 IF o$(1)<"A" THEN  LET o$=o$(2 TO ): GO TO 2290
2300 IF o$(LEN o$)<"A" THEN  LET o$=o$( TO LEN o$-1): GO TO 2300
2310 IF o$<>"ORO ORO CHOCOLATE DEL LORO" THEN  PRINT "No son las palabras magica
s     correctas.": RETURN
2320 IF e(15,3)<0 THEN  PRINT "Ya lo has hecho antes.": RETURN
2330 PRINT "El Emperador palidece al oir el conjuro.Su cuerpo empieza a des-menu
zarse hasta quedar reducido a polvo.Entre el polvo aciertas a descubrir un diama
nte."
2340 LET e(15,3)=-1
2350 LET e(3,2)=1
2360 RETURN
2370 REM PONER O INSERTAR
2380 IF OBJETO<>3 OR e(3,2)<>1 OR mapa<>25 THEN  PRINT "No puedes."
2390 PRINT "Al introducir el diamante en la cuenca del ojo de la estatua se conv
ierte en la princesa Aifos  de carne y hueso."
2400 LET e(12,3)=1
2410 RETURN
2420 REM  DATA DE VERBOS
2430 DATA "INSE"
2440 DATA "PONER"
```

```
2450 DATA "IR"
2460 DATA "EXA"
2470 DATA "COG"
2480 DATA "DEJ"
2490 DATA "ATAC","MAT"
2500 DATA "ENCE"
2510 DATA "ABR"
2520 DATA "DAR"
2530 DATA "LEER"
2540 DATA "DECIR"
2550 DATA "*"
2560 REM  DATA PARA LOS OBJETOS
2570 DATA "DAG","LINT","DIAM","FLOT","LLAV","COFR","PERG","ORO"
2580 DATA "PUER","VENT","ARB","EST","ESQU","MAGR","EMPERADOR","BAU","CERR","DRAG
","CAS","CALL"
2590 DATA "*"
2600 REM  DATA PARA MAPA
2610 RESTORE 2670
2620 DIM 1$(30,10)
2630 FOR I=1 TO 30
2640 READ L$(I)
2650 NEXT i
2660 LET MAPA=25
2670 DATA "0007020001"
2680 DATA "0008030201"
2690 DATA "0009000201"
2700 DATA "0010050001"
2710 DATA "0000000403"
2720 DATA "0012000002"
2730 REM SEGUNDA FILA
2740 DATA "0113080001"
2750 DATA "0214000701"
2760 DATA "0315000001"
2770 DATA "0400110001"
2780 DATA "0000001003"
2790 DATA "0618000004"
2800 REM TERCERA FILA
2810 DATA "0700000001"
2820 DATA "0800000001"
2830 DATA "0900160001"
2840 DATA "0000171501"
2850 DATA "0000001603"
2860 DATA "1224000004"
2870 REM CUARTA FILA
2880 DATA "0000000005"
2890 DATA "0026210006"
2900 DATA "0000222007"
2910 DATA "0028232107"
2920 DATA "0000242213"
2930 DATA "1830000004"
2940 REM QUINTA FILA
2950 DATA "0000260008"
2960 DATA "2000272509"
2970 DATA "0000002610"
2980 DATA "2200290011"
2990 DATA "0000002812"
3000 DATA "2400000004"
3010 REM DESCRIPCIONES DE LOS LUGARES
3020 DATA "las cavernas del dragon.Un ligero olor a azufre inunda el ambiente,y
la humedad cala hasta los huesos"
3030 DATA "la sala del trono del Emperador.Enormes candelabros y cortinajes flan
quean el camino alfombrado hasta el trono dorado"
3040 DATA "la guarida de uno de los tres dragones.Aqui el olor a azufre es basta
nte acusado"
3050 DATA "las inmensas dependencias del castillo del Emperador"
3060 DATA "el pozo.Frente a ti,una puerta vieja y desvencijada.Esta tan oscuro q
ue no puedes ver casi ni la cuerda a la que te aferras con todas tus fuerzas"
3070 DATA "el pantano.Aqui los animales son los reyes;no se asustan ante tu pres
encia.Mas aun,te miran con curiosidad"
```

```
3080 DATA "el esrecho camino que conduce al castillo"
3090 DATA "tu casa.Sobre la cama hay una estatua de piedra caliza y blanca.El po
zo del agua tiene la tapdera quitada.Todo parece abandonado,como si hubieras est
ado dormido siglos....."
3100 DATA "la calle de lo que deberia haber sido la populosa ciudad donde habita
bas la noche anterior.En su lugar,un descampado en el que se alza una ridicula y
 misera choza"
3110 DATA "la choza de Magra.Ciertamente no ha sido limpiada desde hace mucho ti
empo.El polvo del ambiente casi se puede cortar con cuchillo,y el agua que hierv
e en el debil fuego contribuye a enrarecer el ambiente"
3120 DATA "el bosque.Hacia el este puedes ver un gran arbol hueco por dentro.Par
a entrar en el,alguien ha abierto una diminuta puerta"
3130 DATA "el interior del arbol.Dentro del arbol se han acomodado varias coloni
as de gusanos,pero todavia se pueden distinguir signos de que alguien estuvo aqu
i hace mucho tiempo"
3140 DATA "el foso del castillo.El agua negra y sucia desprende un olor fetido,p
ero no es momento de delicadezas.Unos cocodrilos salen a tu paso."
3150 DIM E(20,3)
3160 RESTORE 3210
3170 FOR i=1 TO 20
3180 READ e(I,1),e(I,2)
3190 NEXT i
3200 RETURN
3210 DATA 29,1,25,0,6,0,29,1,27,0,17,0,13,1,25,0,19,1,6,1,26,0,25,1,6,0,27,1,6,1
,25,1,19,1,17,1,25,1,26,1
```

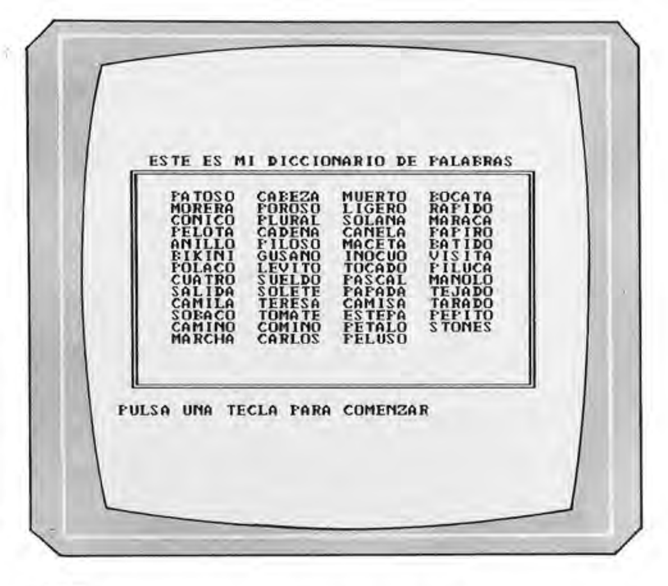
Para terminar, hay que decir que este programa sólo funciona en el SPECTRUM, pero las versiones para los demás ordenadores aparecerán más adelante.

Programa: Mastermind de palabras

El siguiente programa que vamos a ver es una modificación del famoso Mastermind. En esta versión, en vez de tener que adivinar una serie de números, tendremos que ver si podemos sacar una palabra que ha pensado el ordenador. Dicha palabra será una de las que aparecen en el diccionario que el programa lleva incluidas.

Cada vez que nosotros le respondamos al ordenador, éste nos dirá cuántas letras hemos acertado. Si una letra está acertada pero no está colocada en su posición, entonces nos la marcará con una o mayúscula (O). A esta letra se la denomina HERIDO. Si la letra no sólo pertenece a la palabra que tenemos que adivinar, sino que además está colocada en la misma posición que en la palabra, el ordenador nos lo marcará con una x mayúscula (X). A esta letra la llamaremos MUERTO.

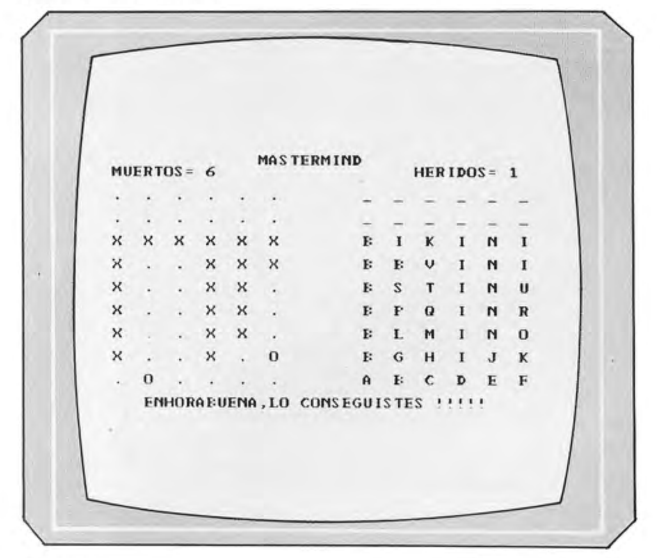
El juego termina cuando hemos acerta-




Diccionario de palabras que contiene el programa.

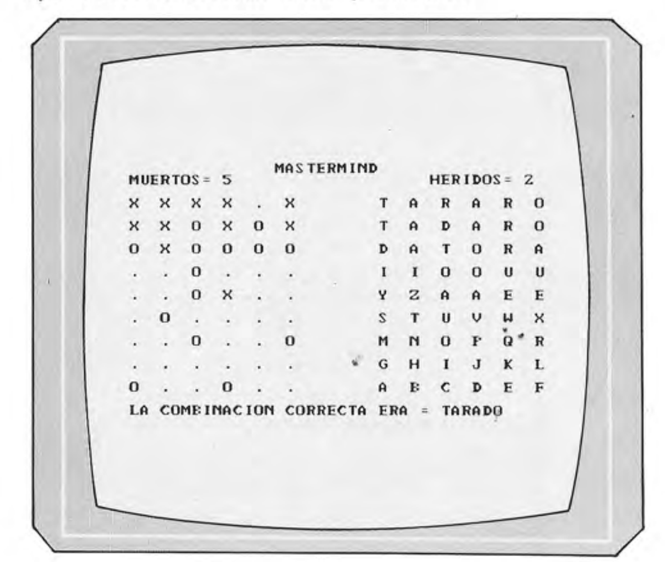
do la palabra o cuando hemos agotado las nueve oportunidades que tenemos para hacerlo.

Si no acertamos la palabra, el ordenador nos dirá cuál era y nos preguntará si queremos echar otra partida.





Una partida ganada.





Si no acertamos, el ordenador nos dice qué palabra era.

```
100 REM ***********************************************************************
101 REM *********************** M A S T E R M I N D *************************
102 REM ***********************************************************************
103 CLS
104 DIM A(6)
105 DIM A$(6)
106 DIM G(6)
107 LET B$="MASTERMIND ED. SIGLO CULTURAL"
108 LET L=LEN(B$)
109 FOR C=1 TO L
110    FOR F=(5+L)-5 TO 5+L
111       LOCATE 10,F
112       PRINT "  ";MID$(B$,L,1)
113    NEXT F
114    LET L=L-1
115 NEXT C
116 LOCATE 21,12
117 PRINT "PULSA UNA TECLA"
118 IF INKEY$="" THEN GOTO 118
119 CLEAR
120 GOSUB 308
121 RESTORE 230
122 CLS
123 RANDOMIZE TIMER
124 FOR F=1 TO INT(RND*50)+1
125    READ P$
126 NEXT F
127 FOR F=1 TO 6
128    LET G(F)=ASC(MID$(P$,F,1))
129    LET A(F)=ASC(MID$(P$,F,1))
130 NEXT F
131 LOCATE 1,15
132 PRINT "MASTERMIND"
133 FOR F=3 TO 20 STEP 2
134    LOCATE F,15
135    PRINT ".  .  .  .  .  .          _  _  _  _  _  _"
136 NEXT F
137 LOCATE 2,1
138 PRINT "MUERTOS="
139 LOCATE 2,30
```

```
140 PRINT "HERIDOS="
141 LET Y1=20:LET X1=25:LET Y2=20:LET X2=-2
142 LOCATE 2,9
143 PRINT HE
144 LOCATE 2,38
145 PRINT MU
146 FOR W=1 TO 9
147    LOCATE 22,1
148    PRINT "PULSA LA SECUENCIA DE LETRAS QUE QUIERAS"
149    FOR F=1 TO 6
150       LET K$=INKEY$
151       IF K$="" THEN GOTO 150
152       IF ASC(K$)<65 OR ASC(K$)>90 THEN GOTO 150
153       LOCATE Y1,X1
154       PRINT K$
155       LET X1=X1+3
156       LET A$(F)=K$
157    NEXT F
158    LOCATE 22,1
159    PRINT SPACE$(40)
160    LET X2=-2
161    FOR F=1 TO 6
162       IF CHR$(A(F))=A$(F) THEN LET MU=MU+1:LOCATE Y2,X2+(F*3):PRINT "X":LET A$(F)=" ":LET A(F)=0:LOCATE 2,9:PRINT MU
163    NEXT F
164    IF MU=6 THEN GOTO 182
165    FOR F=1 TO 6
166       LET X2=-2
167       FOR C=1 TO 6
168          IF CHR$(A(C))=A$(F) AND F<>C AND A(C)<>0 AND A$(F)<>" " THEN LET A(C)=0:LET A$(F)="":LET HE=HE+1:LOCATE Y2,X2+(F*3):PRINT "O":LOCATE 2,38:PRINT HE
169       NEXT C
170    NEXT F
171    LET Y2=Y2-2
172    LET Y1=Y1-2
173    LET X1=25
174    FOR F=1 TO 6
175       LET A(F)=G(F)
176    NEXT F
177    LET HE=0
178    LET MU=0
179 NEXT W
180 GOTO 208
181 REM
182 REM ************************************************************************
183 REM ************* FINAL DEL JUEGO.ADIVINADAS LAS 6 LETRAS ***************
184 REM ************************************************************************
185 REM
186 LOCATE Y1,25
187 LET C=1
188    FOR F=25 TO 40 STEP 3
189    LOCATE Y1,F
190    PRINT CHR$(G(C))
191    LET C=C+1
192 NEXT F
193 LOCATE 22,1
194 PRINT SPACE$(40)
195 LOCATE 22,4
196 PRINT "ENHORABUENA,LO CONSEGUISTES !!!!!"
197 FOR F=1 TO 2000
198 NEXT F
199 LOCATE 22,1
200 PRINT SPACE$(40)
201 LOCATE 22,1
202 PRINT "  ( QUIERES JUGAR OTRA PARTIDA (S/N) ?"
203 LET K$=INKEY$
204 IF K$="S" OR K$="s" THEN GOTO 119
205 IF K$="N" OR K$="n" THEN CLS:PRINT "ADIOS ...":END
206 GOTO 203
207 REM
```

```
208 REM *********************************************************************
209 REM ******************** NO ACERTASTE LAS 6 LETRAS ************************
210 REM *********************************************************************
211 REM
212 LOCATE 22,1
213 PRINT SPACE$(40)
214 LOCATE 22,6
215 PRINT "OOOH ,NO LO CONSEGUISTE  !!!"
216 FOR I=1 TO 1000
217 NEXT I
218 LOCATE 22,1
219 PRINT SPACE$(40)
220 LOCATE 22,1
221 PRINT "LA COMBINACION CORRECTA ERA = "P$
222 FOR F=1 TO 1000
223 NEXT F
224 GOTO 199
225 REM
226 REM ******************************************************************
227 REM ************** TABLA DE PALABRAS *********************************
228 REM ******************************************************************
229 REM
230 DATA "PATOSO"
231 DATA "CABEZA"
232 DATA "MUERTO"
233 DATA "BOCATA"
234 DATA "MORERA"
235 DATA "POROSO"
236 DATA "LIGERO"
237 DATA "RAPIDO"
238 DATA "CONICO"
239 DATA "PLURAL"
240 DATA "SOLANA"
241 DATA "MARACA"
242 DATA "PELOTA"
243 DATA "CADENA"
244 DATA "CANELA"
245 DATA "PAPIRO"
246 DATA "ANILLO"
247 DATA "PILOSO"
248 DATA "MACETA"
249 DATA "BATIDO"
250 DATA "BIKINI"
251 DATA "GUSANO"
252 DATA "INOCUO"
253 DATA "VISITA"
254 DATA "POLACO"
255 DATA "LEVITO"
256 DATA "TOCADO"
257 DATA "PILUCA"
258 DATA "CUATRO"
259 DATA "SUELDO"
260 DATA "PASCAL"
261 DATA "MANOLO"
262 DATA "SALIDA"
263 DATA "SOLETE"
264 DATA "PAPADA"
265 DATA "TEJADO"
266 DATA "CAMILA"
267 DATA "TERESA"
268 DATA "CAMISA"
269 DATA "TARADO"
270 DATA "SOBACO"
271 DATA "TOMATE"
272 DATA "ESTEPA"
273 DATA "PEPITO"
274 DATA "CAMINO"
275 DATA "COMINO"
276 DATA "PETALO"
277 DATA "STONES"
```

```
278 DATA "MARCHA"
279 DATA "CARLOS"
280 DATA "PELUSO"
281 DATA " "
282 DATA " "
283 REM
284 REM ***********************************************************************
285 REM ********************** DIBUJO DE VENTANA ****************************
286 REM ***********************************************************************
287 REM
288 LOCATE F1,C1
289 PRINT CHR$(201);
290 FOR I=C1+1 TO C2-1
291    PRINT CHR$(205);
292 NEXT I
293 PRINT CHR$(187)
294 FOR I=F1+1 TO F2-1
295    LOCATE I,C1
296    PRINT CHR$(186)
297    LOCATE I,C2
298    PRINT CHR$(186)
299 NEXT I
300 LOCATE F2,C1
301 PRINT CHR$(200);
302 FOR I=C1+1 TO C2-1
303    PRINT CHR$(205);
304 NEXT I
305 PRINT CHR$(188)
306 RETURN
307 REM
308 REM ***********************************************************************
309 REM ********************** DIBUJO DEL DICCIONARIO *************************
310 REM ***********************************************************************
311 REM
312 CLS
313 LET F1=2:LET C1=2
314 LET F2=20:LET C2=39
315 RESTORE 230
316 GOSUB 288
317 LOCATE 1,4
318 PRINT "ESTE ES MI DICCIONARIO DE PALABRAS"
319 FOR F=4 TO 16
320    READ Q$,W$,E$,R$
321    LOCATE F,6
322    PRINT Q$;"  ";W$;"  ";E$;"  ";R$
323 NEXT F
324 LOCATE 21,26
325 PRINT "PULSA UNA TECLA PARA COMENZAR"
326 IF INKEY$="" THEN GOTO 326
327 RETURN
```

Este programa ha sido realizado en un IBM. Para que funcione en los demás ordenadores hay que realizar los siguientes cambios:

COMMODORE:

```
103 PRINT CHR$(147)
111 POKE 214,9:POKE 211,F-1
116 POKE 214,20:POKE 211,11
118 GET A$:IF A$="" THEN GOTO 118
122 PRINT CHR$(147)
123 LET A=RND(-TI)
124 FOR F=1 TO INT(RND(1)*50)+1
131 POKE 214,0:POKE 211,14
134 POKE 214,F-1:POKE 211,14
137 POKE 214,0: POKE 211,1
```

```
139 POKE 214,1:POKE 211,29
142 POKE 214,1:POKE 211,8
144 POKE 214,1:POKE 211,27
147 POKE 214,21:POKE 211,0
150 GET K$
153 POKE 214,Y1-1:POKE 211,X1-1
158 POKE 214,21:POKE 211,0
159 FOR I=1 TO 40:PRINT " ";:NEXT I
162 IF CHR$ (A(F)) = A $ (F) THEN LET
MU=MU+1:POKE 214,Y2-1:POKE
211,X2-1+(F*3):PRINT "X":LET A$(F)= " ":LET
A(F)=0:POKE 214,1:POKE 211,8:PRINT MU
168 IF CHR$(A(C))=A$(F) AND F<>C AND
A(C)<>0 AND A$(F)<>" " THEN LET
A(C)=0:LET A$(F)="":LET HE=HE+1:POKE
214,Y2-1:POKE 211,X2-1+(F*3):PRINT
"O":POKE 214,1:POKE 211,27:PRINT HE
186 POKE 214,Y1-1:POKE 211,24
189 POKE 214,Y1-1:POKE 211,F-1
193 POKE 214,21:POKE 211,0
194 FOR I=1 TO 40:PRINT"";:NEXT I
195 POKE 214,21:POKE 211,3
199 POKE 214,21:POKE 211,0
200 FOR I=1 TO 40:PRINT " ";:NEXT I
201 POKE 214,21:POKE 211,0
203 GET K$
212 POKE 214,21:POKE 211,0
213 FOR I=1 TO 40:PRINT " ";:NEXT I
214 POKE 214,21:POKE 211,5
218 POKE 214,21:POKE 211,0
219 FOR I=1 TO 40:PRINT " ";:NEXT I
220 POKE 214,21:POKE 211,0
288 POKE 214,F1-1:POKE 211,C1-1
289 PRINT CHR$(176);
291 PRINT CHR$(99);
293 PRINT CHR$(174)
295 POKE 214,I-1:POKE 211,C1-1
296 PRINT CHR$(98)
297 POKE 214,I-1:POKE 211,C2-1
298 PRINT CHR$(98)
300 POKE 214,F2-1:POKE 211,C1-1
301 PRINT CHR$(173);
303 PRINT CHR$(99);
305 PRINT CHR$(189)
312 PRINT CHR$(147)
317 POKE 214,0:POKE 211,3
321 POKE 214,F-1:POKE 211,5
324 POKE 214,20:POKE 211,25
326 GET A$:IF A$="" THEN GOTO 326
```

AMSTRAD:

```
111 LOCATE F,10
116 LOCATE 12,21
123 RANDOMIZE TIME
124 FOR F=1 TO INT(RND(1)*50)+1
131 LOCATE 15,1
134 LOCATE 15,F
137 LOCATE 1,2
139 LOCATE 30,2
142 LOCATE 9,2
144 LOCATE 38,2
147 LOCATE 1,22
153 LOCATE X1,Y1
158 LOCATE 1,22
162 IF CHR$(A(F))=A$(F) THEN LET
MU=MU+1:LOCATE X2+(F*3),Y2:PRINT
"X":LET A$(F)=" ":LET A(F)=0:LOCATE 9,2:
PRINT MU
168 IF CHR$ (A(C))=A$(F) AND F<C AND
(C)<>0 AND A$(F)<>" " THEN LET
A(C)=0:LET A$(F)="":LET HE=HE+1:LOCAT E
X2+(F*3), Y2:PRINT "O":LOCATE 38,2:PRINT
HE
186 LOCATE 25,Y1
189 LOCATE F,Y1
193 LOCATE 1,22
195 LOCATE 4,22
199 LOCATE 1,22
201 LOCATE 1,22
212 LOCATE 1,22
214 LOCATE 6,22
218 LOCATE 1,22
220 LOCATE 1,22
288 LOCATE C1,F1
289 PRINT CHR$(150);
291 PRINT CHR$(154);
293 PRINT CHR$(156)
295 LOCATE C1,I
296 PRINT CHR$(149)
297 LOCATE C2,I
298 PRINT CHR$(149)
300 LOCATE C1,F1
301 PRINT CHR$(147);
303 PRINT CHR$(154);
305 PRINT CHR$(153)
317 LOCATE 4,1
321 LOCATE 6,F
324 LOCATE 1,26
```

MSX:

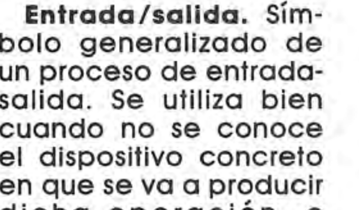
Las modificaciones para el MSX son las mismas que para el AMSTRAD con la excepción de las siguientes líneas:

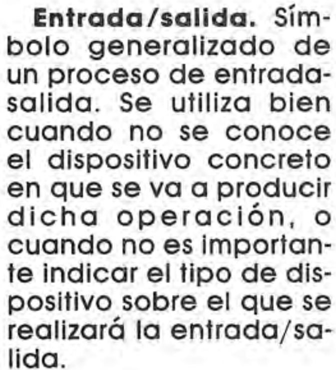
```
123 LET A=RND(-TIME)
289 PRINT CHR$(219);
291 PRINT CHR$(219);
293 PRINT CHR$(219)
296 PRINT CHR$(219)
298 PRINT CHR$(219)
301 PRINT CHR$(219);
303 PRINT CHR$(219);
305 PRINT CHR$(219)
```

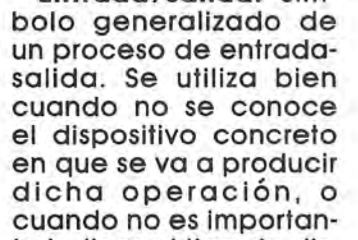
TECNICAS DE ANALISIS

ORGANIGRAMAS (II)

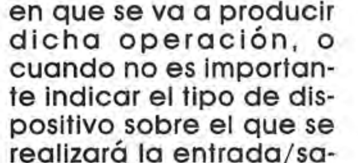


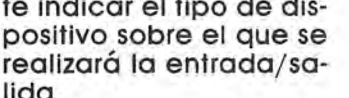




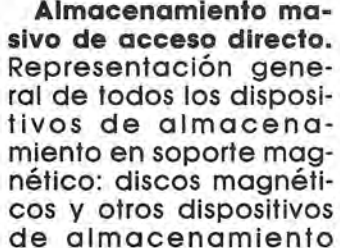


Entrada/salida. Símbolo generalizado de un proceso de entrada-salida. Se utiliza bien cuando no se conoce el dispositivo concreto en que se va a producir dicha operación, o cuando no es importante indicar el tipo de dispositivo sobre el que se realizará la entrada/salida.

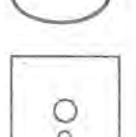




Almacenamiento masivo de acceso directo. Representación general de todos los dispositivos de almacenamiento en soporte magnético: discos magnéticos y otros dispositivos de almacenamiento masivo de datos.



Disco magnético. Cualquier tipo de discos magnéticos (excepto diskettes).



Disco flexible. Específicamente este tipo de disco.





Cinta magnética. No es usual distinguir entre las cintas de gran capacidad en cassette y las cintas en cartucho (cintas tipo «streamer»).



Salida visual por pantalla. Suele aparecer junto al símbolo de entrada manual formando un conjunto representativo de una estación de trabajo (pantalla CRT más teclado de entrada de datos).



Entrada manual de datos desde un teclado.



Documento en general. Normalmente suele ser documento de salida del proceso.



Almacenamiento externo. Suele representarse con este símbolo cualquier almacenamiento externo, si no se desea (o no se puede) especificar el tipo de dispositivo en que está soportado.

Se utiliza también este símbolo (normalmente de tamaño menor) para representar una tarea de intercalación.

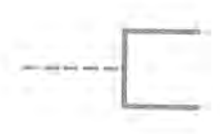
Selección. Representa un proceso de extracción de datos según criterios definidos.

Conector. En ocasiones es necesario reunir varias líneas de flujo o separarlas (especial-

mente en ciertas líneas que representan relaciones de archivos o programas) sin que se desee reseñar ningún proceso especial en ese momento: se utiliza entonces este símbolo.



Clasificación. Se representa con este doble símbolo (como una selección y una intercalación reunidas) el proceso de clasificación de los datos de un fichero.



Comentario. Se suelen incluir comentarios fuera o al margen del organigrama poniendo una línea de puntos en cuyo extremo (dentro de la horquilla) se escribe el comentario.



Indicadores de flujo. Con esas puntas de flecha se suele señalar el sentido del flujo de datos cuando es útil indicarlo (es usual reservar este tipo de información para los organigramas).

En los procesos con ordenadores generales (grandes) de proceso administrativo se suelen utilizar otros dispositivos, que se representan por los siguientes símbolos:



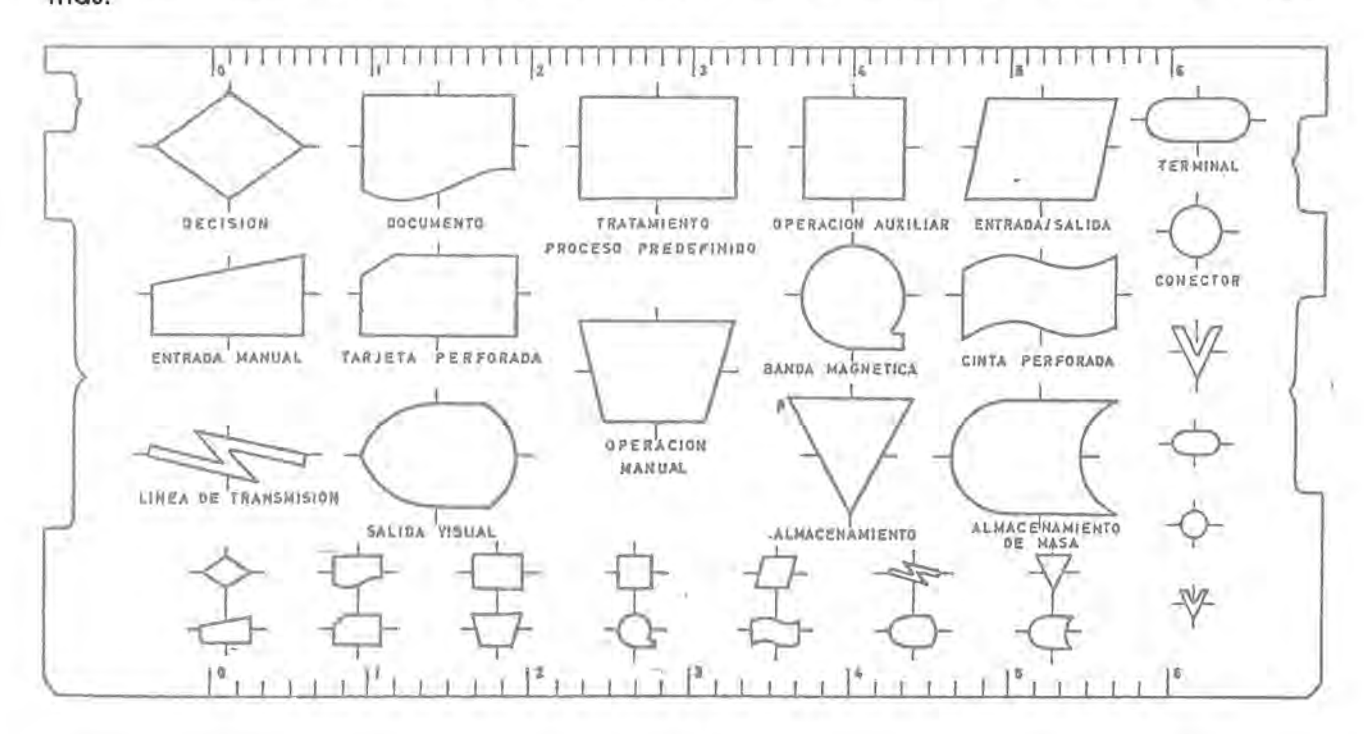
Tarjeta perforada.

Fichero completo de tarjetas perforadas.

Banda perforada.

Registrador de curvas.

En la figura 1 se puede ver una plantilla (que se puede comprar en cualquier establecimiento especializado), con los símbolos que se suelen utilizar en la preparación de organigramas.



TECNICAS DE PROGRAMACION

EXPRESIONES

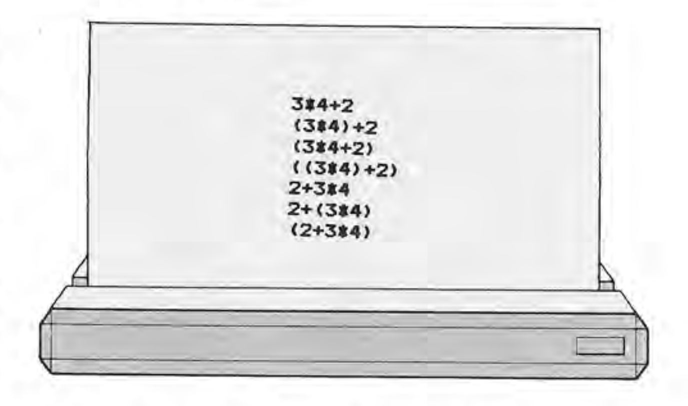


EN el capítulo anterior hablamos de la distinta precedencia que pueden tener las diversas operaciones cuando se mezclan en una expresión única. Sin embargo, en casi todos los lenguajes de programación existe una manera de evitar que dicha precedencia tenga efecto, de manera que el programador siempre podrá conseguir que las operaciones se efectúen en el orden y de la manera que interese. Basta, para ello, con utilizar adecuadamente los paréntesis.

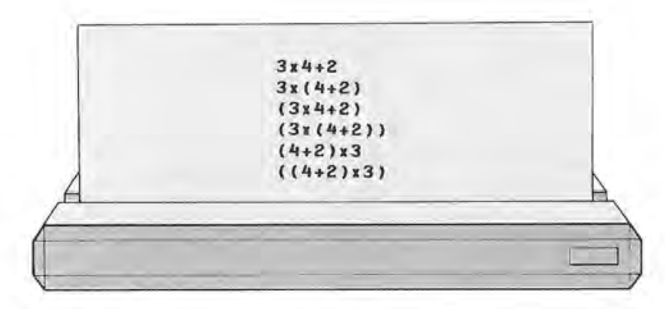
Hemos visto que, en PASCAL y en BASIC, la expresión 3* 4 + 2, dará un resultado de 14, pues la multiplicación tiene precedencia respecto a la suma. Vimos también que en APL, donde las reglas son diferentes, la expresión equivalente 3 × 4 + 2 da un resultado de 18, pues las operaciones se efectúan en este lenguaje comenzando por la que está situada más a la derecha. Pero ¿acaso no podremos conseguir que esta misma combinación de operaciones nos dé en BASIC o PASCAL el resultado 18, o en APL el resultado 14, invirtiendo de alguna manera el orden natural de las prioridades?

Pues, en efecto, es posible. La expresión BASIC (o PASCAL) 3* (4 + 2) encierra entre paréntesis la operación suma, junto con sus dos sumandos, para indicarle al intérprete o al compilador que la suma debe efectuarse en este caso antes que la multiplicación por 3. De igual manera, la expresión APL (3 × 4) + 2 indica al intérprete que ahora es la multiplicación la que tiene precedencia. En este lenguaje se habría obtenido el mismo resultado, sin necesidad de utilizar paréntesis, escribiendo la expresión de esta manera: 2 + 3 × 4. Pero esta inversión del orden no es siempre posible, mientras que los paréntesis sí pueden situarse en cualquier lugar de las expresiones.

De hecho, para el principiante puede ser recomendable utilizar muchos paréntesis, incluso aquellos que pueden no ser necesarios, pues así se asegura de que las instrucciones se ejecutarán en el orden exacto que él desea, evitando errores imprevistos y difíciles de descubrir, que pueden deberse a que ha tratado de simplificar sus expresiones utilizando un orden de precedencia de operaciones equivocado. Pues en todos los lenguajes de programación existen muchas maneras diferentes de representar la misma expresión. Por ejemplo, todas las expresiones BASIC (o PASCAL) siguientes son equivalentes:



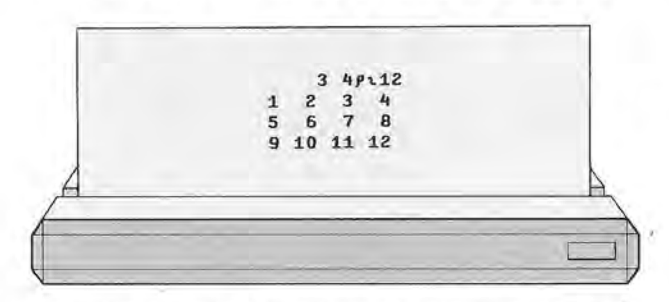
e igualmente lo son las siguientes expresiones APL:



Tipos de expresiones

Hemos visto que las expresiones son fórmulas que contienen una o más operaciones que actúan sobre uno o más datos para producir un dato nuevo o resultado. En capítulos anteriores vimos que los datos tienen dos características propias: estructura y tipo. Pues bien: en la mayor parte de los lenguajes de programación, las operaciones o funciones no pueden actuar sobre la estructura de los datos. En general, con muy raras excepciones, el dato sobre el que se aplican las operaciones y el dato producido como resultado tienen que ser únicos: escalares. Por esta razón, cuando se desea actuar sobre estructuras más complejas (series, tablas, etc.) es preciso utilizar bucles.

Una excepción importante a esta regla es el lenguaje APL, que se basa, precisamente, en un enfoque totalmente contrario. Aquí, la mayor parte de las operaciones fundamentales del lenguaje actúa sobre datos con estructura: escalares, sí, pero tamién vectores, matrices, o entidades de orden superior. Y existen operaciones especiales que actúan sólo sobre la estructura de los datos, sin tocar sus valores. Ya hemos visto algún ejemplo (la función «rho») al hablar de cómo se formaban en APL tablas y series de datos:



Pero lo que sí es importante en todos los lenguajes de programación es el tipo de datos sobre los que actúan las funciones u operaciones primitivas. Estas pueden clasificarse en varios grupos, dependiendo precisamente del tipo de los datos a los que se aplican y del tipo de resultado que producen. Veamos algunos de los grupos más importantes:



Funciones numéricas

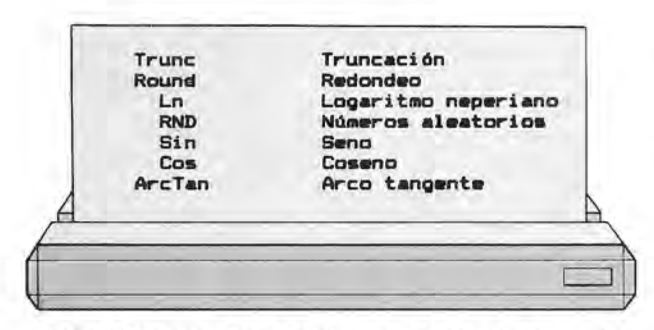
Llamaremos funciones u operaciones numéricas a las que se aplican a datos numéricos para producir resultados numéricos. Las más conocidas són, como es natural, las operaciones aritméticas elementales: suma, resta, multiplicación y división, que los diversos lenguajes representan con símbolos más o menos semejantes a los que se utilizan en las Matemáticas ordinarias. Pero puede haber muchas más, algunas de las cuales reciben nombres en lugar de símbolos.

Veamos, por ejemplo, algunas de las funciones numéricas del lenguaje BASIC:

SIMBOLO O NOMBRE	OPERACION
+	Suma
-	Resta, cambio de signo
*	Multiplicación
/	División
\	División entera
MOD	Resto de dividir
^	Potenciación
SQR	Raiz cuadrada
ABS	Valor absoluto
EXP	e elevado a
INT	Parte entera
LOG	Logaritmo neperiano
RND	Números aleatorios
SIN	Seno
COS	Coseno
ATN	Arco tangente

Veamos también algunas funciones de PASCAL:

SIMBOLO O NOMBRE	OPERACION
+	Suma
-	Resta, cambio de signo
*	Multiplicación
/	División
div	División entera
mod	Resto de dividir
Sqr	Cuadrado
Sqrt	Raiz cuadrada
Abs	Valor absoluto
Exp	e elevado a



Trunc	Truncación
Round	Redondeo
Ln	Logaritmo neperiano
RND	Números aleatorios
Sin	Seno
Cos	Coseno
ArcTan	Arco tangente

Por último, he aquí una lista de funciones numéricas en lenguaje APL:

SIMBOLO	OPERACION DIADICA	OPERACION MONADICA
+	Suma	Identidad
-	Resta	Cambio de signo
×	Multiplicación	Signo
÷	División	Inverso
|	Resto dividir	Valor absoluto
*	Potencia	"e" elevado a
⌊	Mínimo	Parte entera
⌈	Máximo	Techo
⍟	Logaritmo	Log. neperiano
○	Funciones trigonométricas	"pi" por
!	Números combinatorios	Factorial y Gamma de Euler
?	Aleatorios sin reemplazamiento	Aleatorios con reemplazamiento
⌹	Solución de sistemas de ecuaciones	Matriz inversa

Todas las funciones aritméticas APL tienen dos formas: diádica, en la que actúan sobre dos términos (como en 4-X) y monádica, cuando actúan sobre un solo término (como en -X). De esta manera, con el mismo número de símbolos se puede representar un número doble de operaciones aritméticas. Obsérvese también que el APL es uno de los pocos lenguajes de programación que utilizan los símbolos clásicos de las Matemáticas para las cuatro operaciones fundamentales, reservando el asterisco para la potenciación, que al utilizar superíndices no puede representarse fácilmente en un teclado de ordenador.

Funciones lógicas

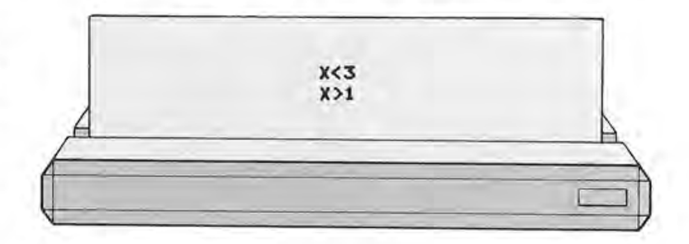
En capítulos anteriores hemos mencionado someramente el tipo de datos lógicos, indicando que se hablaría de él con detalle algo más adelante. Pues bien: ha llegado el momento.

En cualquier lenguaje de programación, se llama «dato lógico» a aquel que tan sólo puede tomar dos valores: «verdadero» y «falso». Se nos presenta con frecuencia este tipo de datos cuando realizamos comparaciones entre los valores de otros dos datos cualesquiera. Por ejemplo: la expresión X=2 puede significar, en muchos lenguajes de programación, que deseamos saber si el valor de la variable X es igual a 2 o no. En el primer caso, decimos que X=2 es verdadero. En el segundo (por ejemplo, si X resulta ser, después de todo, igual a 3) diremos que X=2 es falso.

Veamos una lista de las operaciones de comparación, que actúan sobre datos de cualquier tipo y producen un resultado lógico (verdadero o falso), en los tres lenguajes de programación que estamos considerando: BASIC, PASCAL y APL:

COMPARACION	BASIC	PASCAL	APL
Igualdad	=	=	=
Desigualdad	<> o ><	<>	≠
Mayor que	>	>	>
Menor que	<	<	<
Mayor o igual	>= o =>	>=	≥
Menor o igual	<= o =<	<=	≤

Con los datos lógicos se pueden realizar ciertas operaciones diferentes de las que afectan a los datos numéricos o literales. Supongamos, por ejemplo, que tenemos dos datos lógicos, como los resultados de las expresiones siguientes:



La primera dará el resultado «verdadero» siempre que el valor de la variable X sea menor que 3 y el resultado «falso» en caso contrario. La segunda dará el resultado «verdadero» cuando el valor de X sea mayor que 1 y «falso» cuando sea menor o igual que 1. Para cada valor de la variable X, la expresión X<3 tomará

cierto valor, mientras que X>1 tomará otro que puede o no ser distinto al anterior. Veamos algunos ejemplos.

VALOR DE X	X<3	X>1
⁻1	verdadero	falso
0	verdadero	falso
1	verdadero	falso
2	verdadero	verdadero
3	falso	verdadero
4	falso	verdadero

Se observará que, en algunos casos, las dos expresiones son verdaderas a la vez, mientras que en otros una es verdadera y la otra falsa. También puede ocurrir (aunque no en este ejemplo) que las dos fueran falsas a la vez.

Muchas veces, mientras programamos, puede interesarnos saber si ocurre una de esas coincidencias. Por ejemplo, si las dos comparaciones anteriores se cumplen simultáneamente, lo que significa que el valor de X está comprendido entre 1 y 3 (excluyendo los valores 1 y 3). Si X es una variable que puede adoptar valores con decimales, esto es equivalente a decir que nos interesa saber si su valor está comprendido entre 1.0000001 y 2.9999999, ambos inclusive (suponiendo que trabajamos con ocho cifras decimales de precisión).

Pues bien, todos los lenguajes de programación incluyen algún tipo de operación que actúa sobre valores de tipo lógico y que nos indica cuándo se producen estas coincidencias. Las operaciones lógicas fundamentales son las siguientes:

— Negación, que actúa sobre un dato lógico único. Si el dato es verdadero, su negación es falsa. Si el dato es falso, su negación es verdadera.

— Conjunción, que actúa sobre dos datos. Si los dos son verdaderos, su conjunción es verdadera. Si al menos uno de los dos es falso, su conjunción será falsa. La conjunción se representa comúnmente por la palabra Y o algún término equivalente.

— Disyunción, que actúa sobre dos datos. Si al menos uno de los dos es verdadero, la disyunción es verdadera. Si los dos son falsos, la disyunción es falsa. La disyunción se representa comúnmente por la palabra O o algún término equivalente.

— Disyución exclusiva, que actúa sobre dos datos. Si uno solo es verdadero, la disyunción exclusiva es verdadera. Si los dos son falsos o los dos son verdaderos, la disyunción exclusiva es falsa.

— Equivalencia, que actúa sobre dos datos. Si los dos son verdaderos o los dos son falsos, la equivalencia es verdadera. Si uno es verdadero y el otro falso, la equivalencia es falsa. La equivalencia se representa comúnmente por la palabra IGUAL o algún término equivalente.

— Implicación, que actúa sobre dos datos. Si el primero es verdadero y el segundo es falso, la implicación es falsa. En todos los demás casos, la implicación es verdadera.

Existen otras operaciones lógicas posibles con dos datos lógicos, en cuya descripción no vamos a entrar aquí. En general, todas ellas pueden definirse en función de las que ya hemos visto.

Veamos una lista de las operaciones lógicas de los lenguajes BASIC, PASCAL y APL:

OPERACION	BASIC	PASCAL	APL
Negación	NOT	NOT	~
Conjunción	AND	AND	∧
Disyunción	OR	OR	∨
Disyunción exclusiva	XOR	XOR	≠
Equivalencia	EQV		=
Implicación	IMP		∨~



APLICACIONES

HOJAS DE CALCULO

DE entre los muchísimos tipos de aplicaciones existentes uno de los más conocidos y utilizados es el de las hojas electrónicas (*spreadsheet*). Estas se han convertido desde 1979, fecha de aparición de VisiCalc, en el tipo de programa para microodenadores más ampliamente utilizado. A medida que el progreso electrónico hace crecer el tamaño de las memorias, las hojas electrónicas crecen también y se vuelven más complejas.

 ¿Qué es una hoja de cálculo?

Las hojas electrónicas o de cálculo, así denominadas por la forma que adoptan dentro de la pantalla, son programas concebidos para resolver con facilidad problemas en un amplio campo de aplicaciones numéricas, realizar planificaciones o partiendo de modelos preestablecidos obtener distintos resultados, variando los parámetros de dichos modelos.

La gran cantidad de campos de aplicación de estos programas hacen que gracias a su versatilidad puedan ser utilizados para los más dispares problemas: cálculos financieros, cálculos de estructuras, tablas de valores, presupuestos, control de calidad, etc.

Se trata, en definitiva, de la respuesta a preguntas que, aunque parezca extraño, todavía hoy agobian a directivos, contables, técnicos o al encargado de realizar la planificación familiar: ¿Qué ocurriría si se realiza una inversión mayor de lo previsto?

Para una empresa de reducidas dimensiones, un error en su planificación financiera puede suponer su desaparición del mercado. En una gran empresa, estos errores también se pagan caros, pero los resultados suelen ser menos dramáticos.

En uno y otro caso, la utilización adecuada de una hoja de cálculo puede eliminar muchos problemas.

 Bases de la hoja electrónica

La hoja de datos que tradicionalmente se utiliza en contabilidad consiste en una gran hoja de papel, que suele constar de un par de páginas, dividida en filas horizontales y columnas verticales que forman una malla de celdas o cajas. En muchos aspectos, la hoja de cálculo del ordenador es como la de papel, principalmente sus elementos.

 La hoja electrónica

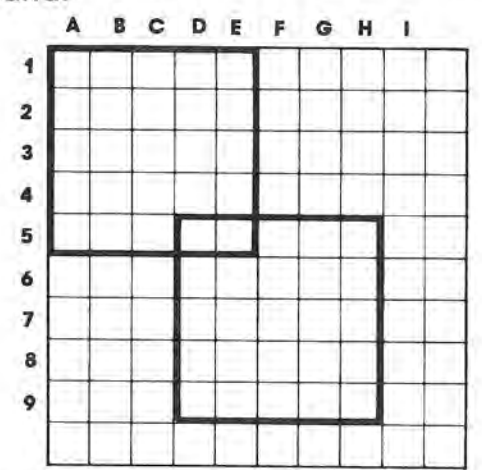
Una vez puesto en marcha el programa nos encontramos con una cuadrícula que, como la de papel, está dividida en filas y en columnas.

Las filas se señalan en una franja vertical, con números consecutivos. Dependiendo de la hoja que hayamos seleccionado, éstos podrán ser negativos o no.

Las columnas, también en función de la hoja elegida, pueden aparecer numera-

das o señaladas con las letras de abecedario, de la siguiente forma: A, B, ...Z, AA, AB, AC...

Evidentemente en la pantalla del ordenador no cabe toda la hoja, por ello en cada momento sólo tendrás una visión parcial de la misma, que se denomina ventana.





Dos ventanas diferentes de una hoja de cálculo.

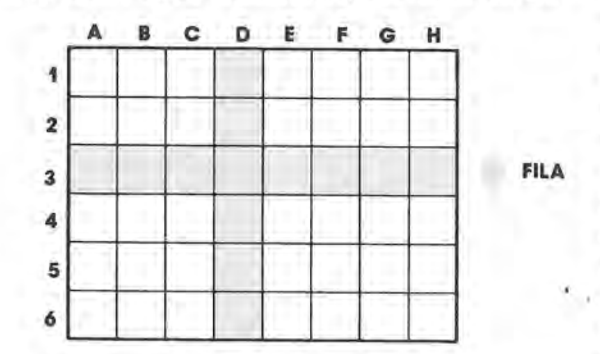
Otra posibilidad que nos ofrecen estos programas son las ventanas múltiples; evidentemente, por cada una de ellas podremos asomarnos a zonas diferentes de la hoja de trabajo que no sean adyacentes.

La ventanas pueden abrirse en vertical, lo cual nos ayuda a comparar columnas, o en horizontal, lo que nos ayuda a comparar filas.



Las celdas

Se definen como la intersección de una fila con una columna; es con ellas con las que vamos a trabajar, y se definen por el número de su fila y la letra/s (o número) de su columna.

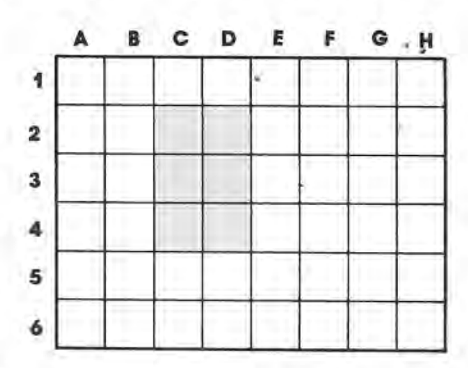




Celda D3, corresponde a la intersección de la fila 3 y la columna D.

CELDA ACTIVA es aquella con la que se está operando o sobre la que se está y se puede actuar; en general, se distingue de las demás porque tiene sobre ella el cursor o puntero de las celdas.

RANGO DE CELDAS es un grupo de celdas que se especifica en un momento dado mediante las coordenadas de la primera y la última de dicho grupo.





Rango de celdas C2: D4.

Las celdas son interactivas entre sí, es decir, pueden relacionarse unas con otras para obtener los resultados deseados.



Contenido de las celdas

LITERALES. Su función es servir para realizar descripciones de otros elementos de la hoja electrónica. La hoja se limita a almacenarlos en la celda correspondiente.

Pueden estar formados por caracteres alfabéticos, símbolos, números o mezclas de ellos, pero hay que tener muy en cuenta que si un número se introduce como alfanumérico no se podrá operar con él.

DATOS NUMERICOS. Representan la información numérica manejada y se pueden utilizar desde otras celdas para operar con ellos.

FORMULAS. Con ellas se obtienen los resultados deseados en la hoja de cálculo, en función de los datos numéricos almacenados. Cuando una celda con dato numérico se varía, el programa se encarga de recalcular las celdas que son función de ella.

Las fórmulas podrán estar formadas por:

— Operadores aritméticos. Son *, /, + y - por este orden de jerarquía.

— Operadores lógicos. Darán como resultado si la expresión es verdadera o falsa y son <)>, <>, <= y >=.

— Otros operadores. Estos dependen de la hoja que hayamos seleccionado, entre los más frecuentes podemos destacar los paréntesis () y el tanto por ciento %.

	MES	ENERO	MES	FEBRERO
MOTIVO	INGRESOS	GASTOS	INGRESOS	GASTOS
SUELDO	150.000		150.000	
ALQUILER		45.000		47.000
DENTISTA		15.000		
OTROS		37.000		40.000
	150.000	97.000	150.000	87.000
TOTAL		53.000		63.000

	LITERAL	LITERAL	LITERAL	LITERAL
LITERAL	LITERAL	LITERAL	LITERAL	LITERAL
LITERAL	NUMERO		NUMERO	
LITERAL		NUMERO		NUMERO
LITERAL		NUMERO		
LITERAL		NUMERO		NUMERO
LITERAL	LITERAL	LITERAL	LITERAL	LITERAL
	FORMULA	FORMULA	FORMULA	FORMULA
LITERAL		FORMULA		FORMULA



Contenido de las celdas.

— Funciones. Pueden utilizarse de forma individual o integradas en fórmulas. Consisten en una amplia gama de pequeños programas cuya necesidad surge a partir de las limitaciones de cálculo que supone la exclusividad de las cuatro operaciones aritméticas básicas.

Suelen subdividirse en tres grupos:

— Funciones matemáticas. Estas a su vez podrán ser para cálculos aritméticos, financieros, trigonométricos, estadísticos...

— Funciones lógicas. Se basan en el álgebra de Boole, permitiendo establecer operaciones condicionales extendiendo notablemente las posibilidades operativas del programa en términos de control de los datos y de los resultados obtenidos. Como resultado no dan valores numéricos, sino lógicos, es decir, verdadero o falso.

— En este tercer grupo de funciones vamos a situar a las que no pertenecen a los otros dos, aunque podrían formar varios grupos a su vez. Son, por ejemplo, las funciones de fecha y hora, las funciones de impresión, funciones de adaptación monetaria, funciones de control...



Los comandos

Hasta ahora conocemos los elementos de la hoja electrónica, pero resulta un misterio su manejo: ¿Cómo le indico de qué tipo de dato se trata? ¿Cómo abrir las ventanas?, o simplemente, ¿cómo iniciar y terminar una sesión de trabajo?

La respuesta a estas preguntas se encuentra en los comandos.

Los comandos son las órdenes que le vamos dando al programa para que realice nuestra hoja de trabajo; pueden aparecer de dos formas:

— Hojas sin menú. En éstas se activan mediante un carácter (como, por ejemplo, /) que el ordenador interpreta para saber que lo que viene a continuación es un comando.

— Hojas con menú. Lo más frecuente es seleccionar el comando correspondiente de este menú, bien pulsando la inicial del mismo, o bien situando el cursor sobre él y pulsando INTRO.

Entre los comandos más comunes se encuentran:

— Comando para la selección de datos alfanuméricos.

— Comando para clasificar por filas o columnas.

— Comando para formatear las celdas: dar la longitud adecuada, centrar o no los datos, especificar número de decimales...

— Comando para insertar o copiar una línea o una columna.

— Comando para crear ventanas.

— Comando para imprimir resultados.

— Comando para recalcular las fórmulas con nuevos datos.

PASCAL



El tipo real



EN varias ocasiones anteriores hemos hablado del tipo REAL, que es el que se utiliza para manejar números que puedan ser no enteros o que puedan estar fuera de los límites establecidos para los de tipo INTEGER. En otras palabras, es el tipo al que habrá que acudir siempre que tratemos con números muy grandes o con decimales; en todos los demás casos convendrá utilizar el tipo INTEGER, pues precisa menor cantidad de memoria y el ordenador es capaz de hacer operaciones con ellos a mucha mayor velocidad.

Pasemos a ver sus principales características.



Precisión y magnitud de los números REAL

Los núme os reales se guardan en la memoria de los ordenadores descomponiéndolos en dos números enteros, y esto es así con la mayor parte de los lenguajes de programación.

Por ejemplo, como 21550000 es igual a 0,2155 multiplicado por 10 elevado a 8 (o sea, por 100000000, un 1 y 8 ceros), podríamos guardar, por un lado, 2155 y, por otro, 8. Al primero de estos números se le denomina MANTISA y al segundo EXPONENTE. Más ejemplos:

- -123,45 es igual a -0,12345 por 1000, luego se guardaría −12345 y 3.
- 0,000256 es igual a 0,256 dividido entre 1000; en este caso se guardaría 256 y -3, indicando el exponente negativo que lo que hay que hacer para obtener el número es dividir la mantisa por 1 y tantos ceros como indique su valor absoluto.

Es decir, se corre la coma tantos lugares como sea necesario para que el valor absoluto del número quede lo más próximo posible a 1, pero por debajo, y a eso se le llama mantisa. Al número de posiciones que se haya tenido que correr la coma se le llama exponente, indicando su signo para dónde hubo que correrla.

En realidad, la forma de guardar los números de tipo REAL es ligeramente distinta, pues los ordenadores utilizan números binarios, esto es, con ceros y unos solamente, pero el fundamento es el mismo. Por supuesto, todo esto se hace de manera automática sin que nosotros nos enteremos. A este sistema de almacenamiento en memoria se le denomina de «coma flotante».

Como consecuencia de esta forma de guardar los números de tipo REAL suceden varias cosas:

1. Cada número REAL se descompone en DOS números y, por tanto, no pueden tener números ordinales asociados. Así, las funciones PRED, SUCC y ORD no se pueden aplicar a valores REAL, ni se pueden utilizar variables REAL para controlar bucles FOR. En otras palabras, el tipo REAL no es un tipo escalar.

No obstante, SI es posible comparar valores REAL entre sí como se hacía con los INTEGER, y el tipo de una función puede ser REAL.

2. Los dos números en que se descomponen, o sea, la mantisa y el exponente,

tienen límites que dependen de cada compilador. Estos límites tienen cierta importancia por lo siguiente:

— La limitación de la mantisa supone una disminución de la precisión con que se pueden guardar los números.

Supongamos que hay que guardar, por ejemplo, el resultado de dividir 100 entre 3, que vale 33,333... con infinitos treses y que es igual a 0,33333... por 100; ahora bien, si la mantisa pudiera tener como máximo 6 cifras, sólo podríamos guardar 333333 y 2 que, en realidad, es lo que corresponde a 33,3333.

La diferencia entre el número realmente guardado y el que debería haber sido se denomina error de redondeo. Así, si dividiéramos 100 entre 3 y lo multiplicáramos luego por 3 obtendríamos 99,9999 en lugar de recuperar el valor original de 100, que es lo esperable en principio.

Como los números se guardan en binario, esta situación se puede dar con números que aparentemente no necesiten muchas cifras, pues, por ejemplo, aunque 0,2 sólo necesita una cifra, en binario 0,2 se escribe como 0,001100110011... Si, por ejemplo, sólo se guardaran 12 dígitos binarios, el número que realmente tendríamos sería 0.199951171875.

Por todo esto, es peligroso comparar dos números reales para ver su igualdad, pues debido al error de redondeo, números que esperábamos que fueran iguales podrían resultar distintos. No obstante, normalmente se pueden esperar al menos 7 u 8 cifras de precisión, e incluso se puede llegar con algunos compiladores a tener 16 cifras o más; esto se debe consultar en el manual de nuestro compilador.

— La limitación del exponente supone, por un lado, un límite superior para la magnitud de los números REAL que se pueden guardar en memoria. Por ello, si el máximo exponente fuera 37 (caso corriente), el número 5432100... (hasta 50 ceros), igual a 0,54321 por 10 elevado a 55, estaría fuera de rango.

Por otro lado, supone que los números muy pequeños se redondearán a cero al ser guardados en memoria. Por ejemplo, 0,00...(50 ceros)...1254 es igual a 0,1254 entre 10 elevado a 50, y si el límite inferior fuera -37, el número se redondearía a cero.



Constantes y variables de tipo REAL

Como puede imaginarse, las constantes de tipo REAL deben ser claramente diferenciables de las de tipo INTEGER para que el compilador sepa cómo guardarlas en memoria.

Si para utilizar en expresiones de tipo REAL necesitáramos un valor, por ejemplo, 15, no podríamos escribirlo tal cual, pues parecería INTEGER.

Para distinguirse, las constantes REAL deben tener punto decimal o exponente o ambos (en inglés, para separar las cifras decimales de las demás se utiliza un punto en lugar de una coma). El exponente se separa del resto del número por medio de la letra E. Veamos unos ejemplos para aclarar esto:

· 15.0	DEBE haber cifras a ambos lados del punto.
· 3E3	3E3 significa 3 por 10 elevado a 3, o sea, 3000.
· -3.17E-4	Equivale a -3,17 entre 10000, o sea, -0,000317.
· 0.33333333333	Problablemente no se guarden todas las cifras.

Las constantes REAL, como las INTEGER, pueden declararse al principio del programa:

```
const Pi = 3.141592653589;
```

Esta constante en concreto se encuentra ya definida con este mismo nombre en muchas versiones de PASCAL.

También las variables se definen como las INTEGER, pero usando la palabra reservada REAL:

```
Var Angulo, Peso: real;
```

Sin embargo, y debido a la carencia de ordinales, no se pueden definir subrangos del tipo REAL.

La asignación de datos a una variable se hace de manera análoga:

```
Angulo := Pi;
Peso := 60.0;
```

Si escribiéramos, por ejemplo, Peso:= 60, al ser 60 una constante de tipo INTEGER, como la variable de destino es REAL, se tiene que producir una transformación de la forma en que se guarda el número 60 (sin que nosotros nos apercibamos de ello). Esto se lleva tiempo de cálculo, por lo que es conveniente ser metódico y escribir constantes reales siempre que corresponda.



Expresiones REAL

Las expresiones de tipo REAL son aquellas en que aparecen constantes, predeclaradas o no, y variables de tipo REAL combinadas entre sí mediante los operadores +, −, * y /.

Son, en principio, muy parecidas a las de tipo INTEGER. El operador DIV no es utilizable aquí, pero en su lugar se tiene el operador / para hacer divisiones, ahora sí, con decimales; el orden de cálculo es similar:

36.0 / (12.5 − 4.5) * 3.0 dará el resultado REAL 13.5

En una expresión REAL pueden aparecer constantes o variables INTEGER, cuyo valor es convertido a tipo REAL automáticamente antes de ser utilizado. En expresiones como:

3 + I * I / 3

donde I fuera una variable o constante entera, aunque todos los elementos son INTEGER, al descubrirse durante el cálculo el operador /, como es exclusivo de los REAL, se pasa a utilizar el tipo REAL, y el resultado será REAL. Este cambio sobre la marcha puede dar lugar a situaciones curiosas que deben ser analizadas con cuidado. Por ejemplo, la expresión:

1000 * 1000 / 10000

aunque su resultado es 100, como se empieza a calcular por la izquierda, produce un error, pues 1000 (INTEGER) por 1000 (INTEGER) da un valor fuera de rango, ya que todavía no se ha descubierto que es una expresión REAL. Si escribiéramos:

1000 / 10000 * 1000 ó 1000 * 1000.0 / 10000, etc.

se resolvería el problema, pues antes de hacer la operación causante del error se pasaría a utilizar el tipo REAL.



Conversión de valores REAL a INTEGER

Ya se ha visto que la conversión de INTEGER a REAL se produce de manera automática; por ello, si I fuese una variable o constante INTEGER cuyo valor quisiéramos guardar en la variable de tipo REAL R, bastaría con escribir R:= I.

Sin embargo, a la inversa hay que definirlo de manera explícita para así poder elegir entre dos posibles modos de conversión:

— La función Round (R), donde R es una variable, constante o expresión REAL, da un resultado INTEGER lo más próximo posible al valor de R:

  Round (14.28) equivale a poner 14, y
  Round (14.59) equivale a poner 15,

  de ahí su nombre («redondea»).

— La función Trunc (R) da un resultado INTEGER igual al número entero más cercano por debajo:

  Trunc (7.3) es igual a 7 y
  Trunc (7.99) también.

Por supuesto, si el valor resultante estuviera fuera de los límites INTEGER admisibles se produciría un error. Nótese que Round (R) es igual a Trunc (R+0.5).



Funciones de librería

El PASCAL proporciona varias funciones matemáticas de tipo REAL predefinidas. Como tales funciones, el parámetro o valor al cual se quiere aplicar la función se debe escribir entre paréntesis a continuación del nombre de la función. Devuelven un valor de tipo REAL y, por tanto, se pueden incorporar a cualquier expresión de este tipo. Por supuesto, el parámetro puede ser una expresión REAL cualquiera (es decir, el paso de parámetros es por valor y no por nombre). Son las siguientes:

FUNCION	SIGNIFICADO	ejemplos	
(•) ABS	valor absoluto	abs (-3.5)	es igual a 3.5
(•) SQR	cuadrado	sqr (2.0)	es igual a 4.0
SQRT	raíz cuadrada	sqrt (6.25)	es igual a 2.5
EXP	exponencial	exp (1.0)	es igual a 2.718281
LN	logaritmo natural	ln (1.0)	es igual a 0.0
SIN	seno	sin (Pi/2.0)	es igual a 1.0
COS	coseno	cos (Pi)	es igual a -1.0
ARCTAN	arcotangente	arctan (1.0)	es igual a Pi/4.0

Las diferentes versiones de PASCAL pueden tener más funciones predefinidas.

Se pueden producir errores durante el cálculo de algunas de estas funciones si el valor del parámetro se sale de los límites establecidos para cada función:

sqrt (-2.0) no es calculable.

Una protección en ese caso sería escribir sqrt (abs(R)), donde R fuese la variable, constante o expresión REAL cuya raíz cuadrada se quisiera obtener.

El parámetro puede ser de tipo INTEGER, pues se convertirá automáticamente en REAL excepto en las funciones marcadas con (•), que proporcionan un resultado del mismo tipo que el parámetro.



Comparaciones entre valores de tipo REAL

Al igual que con el tipo INTEGER, éstas son las comparaciones:

> , < , >= , <= , = , <>

Debe tenerse en cuenta el comentario anterior sobre las comparaciones de igualdad y desigualdad:

100.0 / 3.0 • 3.0 = 100.0

podría dar como resultado FALSE.

Una forma de arreglar esto puede ser poner como condición de igualdad de dos valores el que su diferencia sea menor que una cierta cantidad:

abs (A-B) < DiferenciaMaxima

donde A y B son los valores a comparar.

Normalmente los compiladores admiten comparar valores de tipo REAL con valores de tipo INTEGER, pues éstos son convertidos a REAL antes de hacer la comparación.



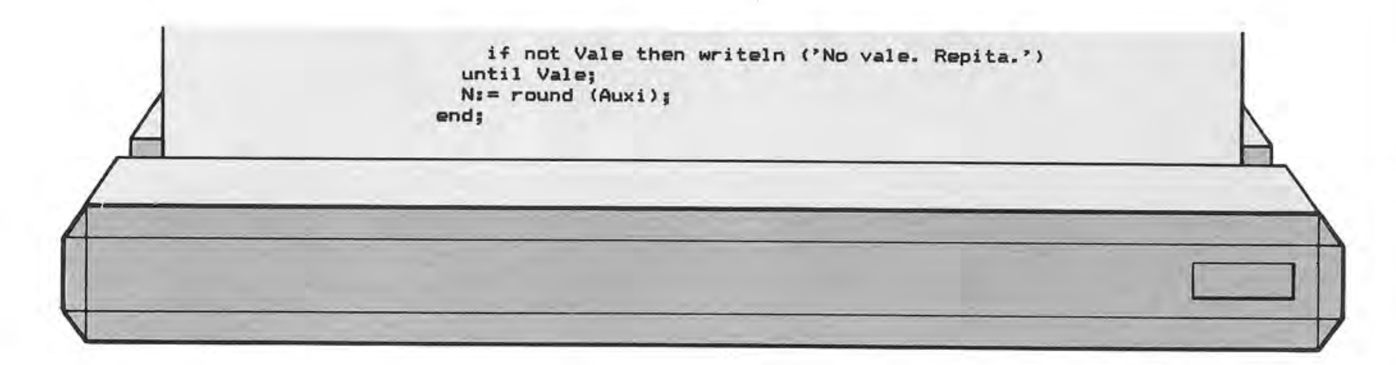
Entrada y salida de datos de tipo REAL

Las variables REAL pueden tomar el valor que se introduzca desde teclado por medio de las instrucciones READ y READLN, que se utilizan de la misma manera que con las variables INTEGER. Valores tecleados aceptables son 3, 72.7, 0.727E2 y 727E-1.

Como ejemplo, veamos un procedimiento para evitar que al leer una variable de tipo INTEGER se pueda producir un error por salida de rango. Utilizaremos una variable REAL para recoger el número tecleado y sólo lo transferiremos a la de tipo INTEGER tras haber comprobado que tiene un valor aceptable:

```
procedure LeeEntero (var N: integer);
 (* lee número entero de teclado cuidando su tamaño *)

   var
     Auxi: real;
     Vale: boolean;
 begin
   repeat
     write ('Valor = '); readln (Auxi);
       (* mirar si el entero correspondiente es licito: *)
     Vale := (abs(Auxi) <= MaxInt);
```



Por supuesto, también sería posible teclear un número fuera de rango incluso para una variable REAL, pero es poco probable que se hiciera por equivocación.

La presentación por pantalla se hace igual que con los valores INTEGER. Los valores REAL se presentan en notación exponencial, esto es, escribiendo mantisa y exponente separados por la letra E. No obstante, es posible presentarlos con aspecto «normal». Para ello, a continuación de la expresión REAL se indica el número de espacios a ocupar y el número de cifras decimales a presentar separados por dos puntos. Por ejemplo:

```
writeln ( 27.2 , 27.2:6:2, 27.2:10:4 )
```

produciría algo como:

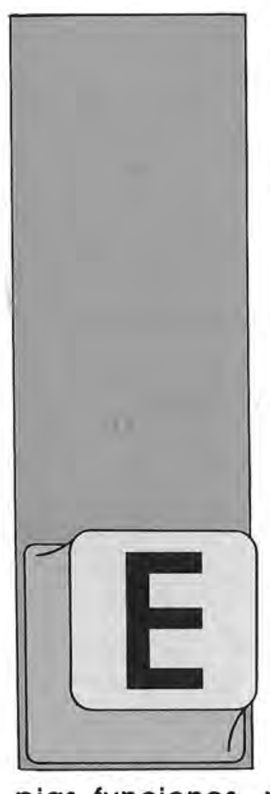
```
2.72000000E+01 27.20    27.2000
```

OTROS LENGUAJES

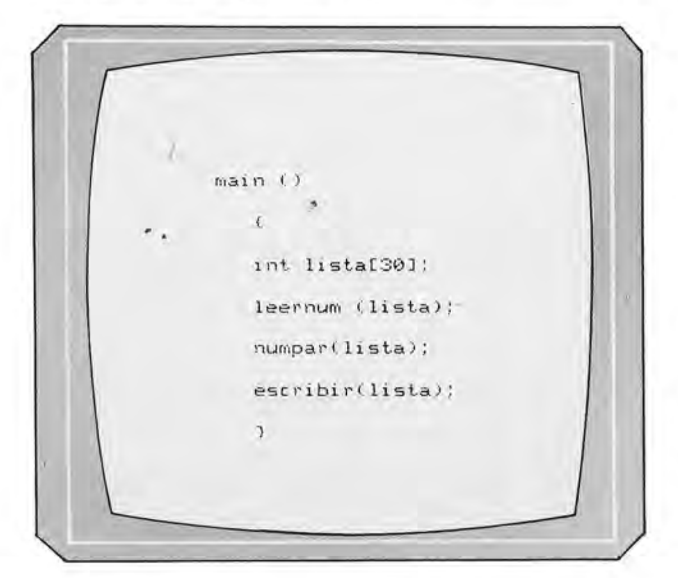
LENGUAJE C



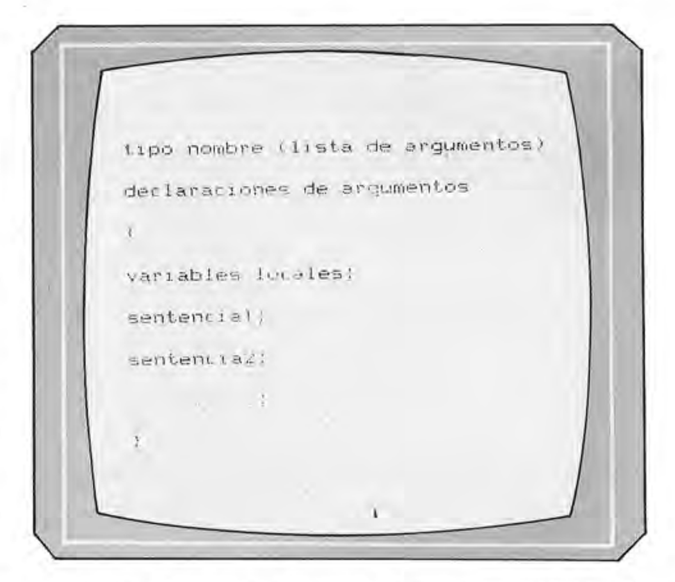
Cómo crear funciones



EL C posee una serie de funciones definidas en el sistema tales como *prinff(), scanff(), getchar(), putchar(), y strlen()*. Nosotros podemos crear nuestras propias funciones, para lo cual estudiaremos el siguiente ejemplo. Supongamos que introducimos por el teclado de nuestro ordenador una serie de números de los cuales queremos obtener los números pares e imprimirlos. Para ello podíamos utilizar un programa como éste:



En el ejemplo anterior deberíamos escribir tres funciones: *leernum, numpar* y *escribir*. Como vemos, dentro del cuerpo principal del programa existen tres llamadas a tres funciones. La forma general que tiene una función es:



Como habrá observado, el nombre de una función nunca va seguido por punto y coma. Deberá tener en cuenta este hecho para saber que está trabajando con una función y no con una sentencia.



Funciones con argumentos

Tipo nombre se refiere al tipo de dato que devuelve la función. Por defecto, se supone que es de tipo *int*.

Lista de argumentos es una relación de nombres separados por comas. Puede no existir ningún argumento, pero sigue haciéndose necesario el empleo de ( ).

En las declaraciones de argumentos declararemos las variables utilizadas como argumentos de la función. Por defecto se suponen que son del tipo *int*, caso de no declararse ninguna.

Variables locales son aquéllas que sólo tienen validez dentro del propio cuerpo de la función, es decir, utilizadas en el programa principal o en otras funciones harían referencia a variables diferentes.

Para devolver un valor al programa de llamada desde una función utilizaremos la palabra clave *return*. Con esta palabra se da por terminada la ejecución de la función, devolviendo control a la siguiente línea de la función de llamada.



Variables externas y alcance de una variable

En el lenguaje de programación C, a diferencia del Pascal, no podemos definir funciones dentro de otras. Las *variables externas* están definidas fuera de la función y pueden ser compartidas por otras, es decir, son variables *globales* y como tales son análogas al COMMON de FORTRAN, por citar un ejemplo.

Si la variable externa se ha definido, cualquier función podrá acceder a ella, con tan sólo referenciarla, evitando en algunos casos grandes listas de argumentos.

Si la variable externa se declara dentro de la función que la utiliza, se deberá declarar con la palabra clave *extern*.

Veamos un ejemplo:

En el ejemplo anterior hemos creado una variable externa *ejemplo* conocida tanto por *main()* como por la función *hola*.

El alcance o validez de una variable externa es su *permanencia*, es decir, permanecen en el ordenador durante la ejecución del programa y al no ser de ninguna función en particular, no quedan eliminadas al finalizar alguna de ellas.

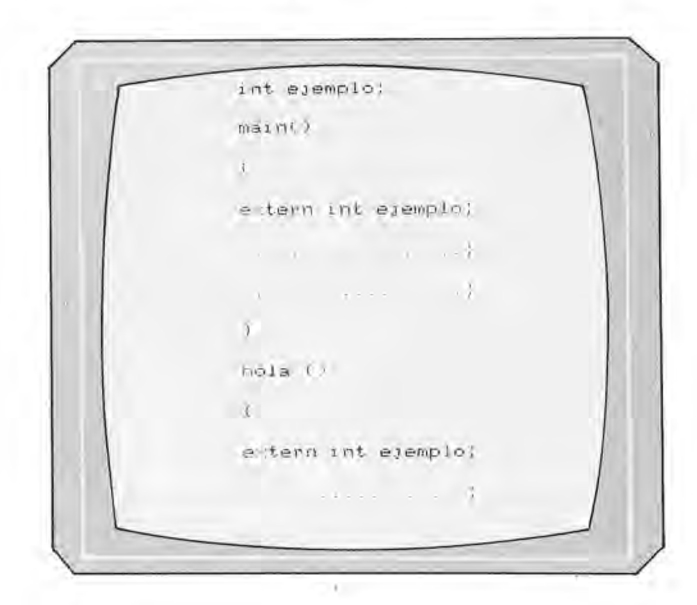


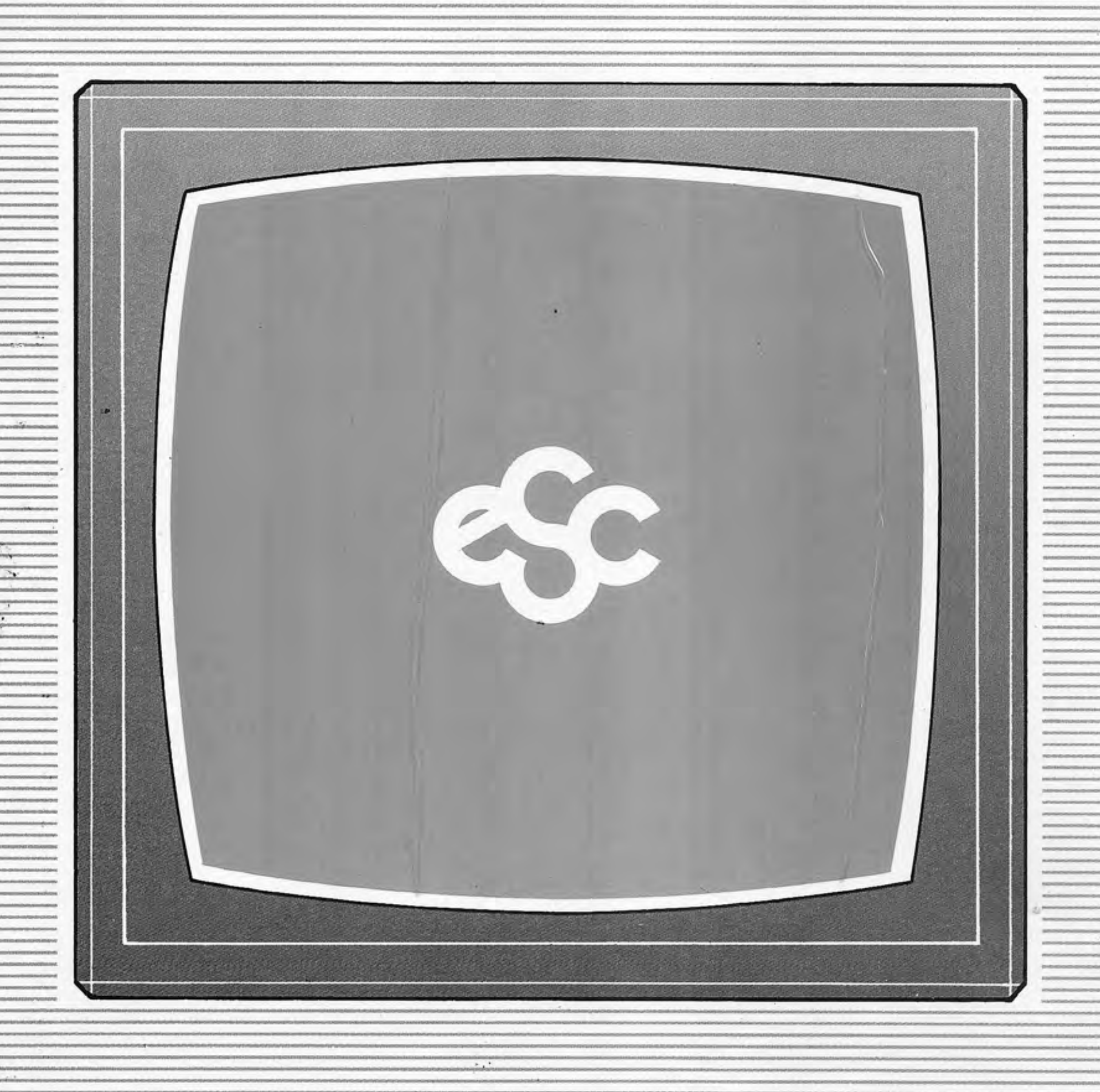
Compilación de programas con varias funciones

La compilación de varias funciones que están en el mismo fichero se lleva a cabo de la misma forma que la compilación de una sola función.

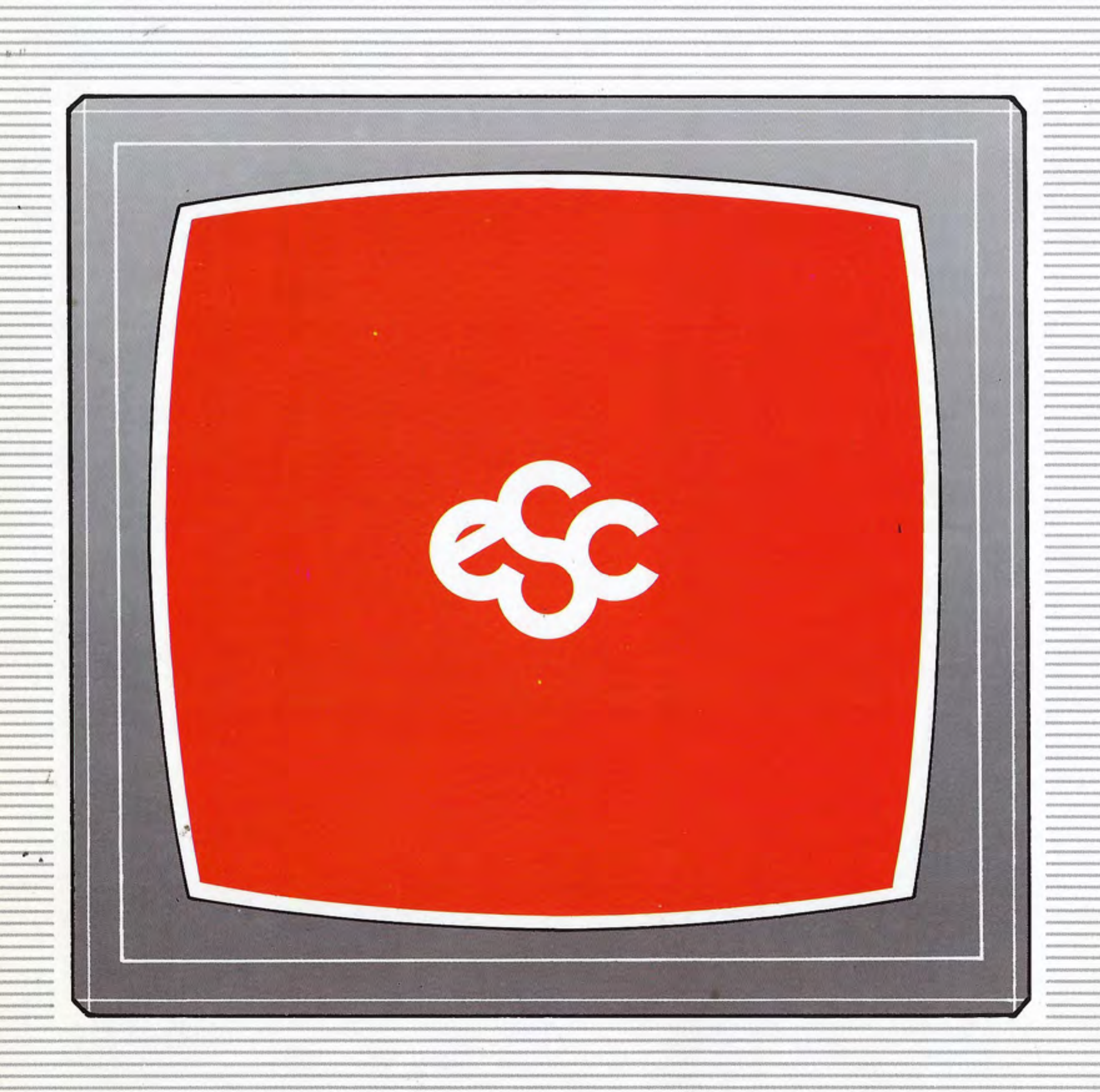
Dependiendo del sistema la compilación se realizará de forma diferente; por ejemplo, en el sistema operativo UNIX el comando *cc prog1.c prog2.c* (suponiendo *prog1.c* y *prog2.c* ficheros que contienen funciones) producirá un ejecutable llamado *a.out*. En otros sistemas tales como *Lattice C* compilando por separado *prog1.c* y *prog2.c* se producirán dos ficheros objetos *prog1.obj* y *prog2.obj*. Estos módulos serán combinados, a través del linker, con los módulos objeto estándar de *c.obj:*

link c prog1 prog2





▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼